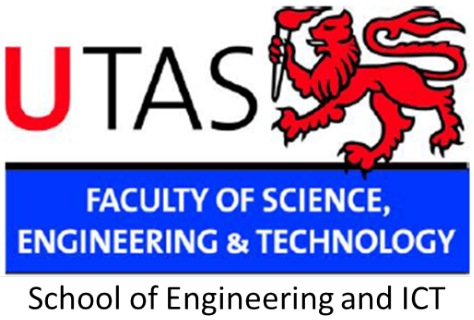

Incremental Knowledge-based System for Schema Mapping

submitted by

Sarawat Anam
B.Sc.(Hons.)

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

University of Tasmania

March 2016

Abstract

Schemas describe the data structures of various domains such as purchase order, conference, health and music. A large number of schemas are available on the Web. Since different schema elements may have the same semantics but exist in distinct schemas, it is important to manage their semantic heterogeneity. Schema matching is usually used to determine mappings between semantically correspondent elements of different schemas. It can be conducted manually, semi-automatically and automatically. Manual matching is a time-consuming, error-prone and expensive process. Fully-automated matching is not possible because of the complexity of the schemas.

This research investigated semi-automatic schema matching systems to overcome manual works for schema mapping. In general, these systems use machine learning and knowledge engineering approaches. Machine learning approaches require training datasets for building matching models. However, it is usually very difficult to obtain appropriate training datasets for large datasets and to change the trained models once mapped. Knowledge engineering approaches require domain experts and time-consuming knowledge acquisition. In order to solve these problems, an incremental knowledge engineering approach - Ripple-Down Rules (RDR) can be a promising approach since it allows its knowledge to grow incrementally. However, acquiring matching rules is still a time-intensive task. In order to overcome the limitations of these independent approaches, a hybrid approach called Hybrid-RDR has been developed by combining a machine learning approach with the Censor Production Rules (CPR) based RDR approach.

First, the most suitable machine learning algorithm, J48 is determined by comparing eleven machine learning approaches including decision trees, rules, Naive Bayes, AdaBootM1, and later combined with CPR based RDR for building Hybrid-RDR approach. This approach constructs a matching model using J48. When new data are

available, the model may suggest incorrect matchings for some cases which are corrected by incrementally adding rules to the knowledge base. The approach reuses the previous match operations (rules) and handles the schema matching problems using an incremental knowledge acquisition process. So users do not need to add, delete or modify schema matching results manually. The Hybrid-RDR approach works for element-level matching that only considers matching names of schema elements. Structure-level matching that considers the hierarchical structure of the schema, is required to adjust incorrect matches found from the element-level matching.

A Knowledge-based Schema Matching System (KSMS) has also been developed that performs element-level matching using Hybrid-RDR and structure-level matching using Similarity Flooding algorithm. This algorithm considers the concept that two nodes are similar when their neighbor elements are similar. The final mappings are generated by combining the results of element-level matching and structure-level matching using aggregation functions. In order to evaluate the performance of the system, evaluations using real world schemas found on the Web have been conducted. Experimental results have shown that the system determines good performance both at element-level matching and structure-level matching. This research has resolved the ongoing problem of elements having different names within different schemas. The KSMS allows for matching of different schemas to produce accurate mappings.

Declaration

This thesis contains no material which has been accepted for a degree or diploma by the University or any other institution, except by way of background information and duly acknowledged in the thesis, and to the best of my knowledge and belief no material previously published or written by another person except where due acknowledgement is made in the text of the thesis, nor does the thesis contain any material that infringes copyright.

Sarawat Anam
June 2016

Authority of Access

The authority of access statement should reflect any agreement which exists between the University and an external organization (such as a sponsor of the research) regarding the work. Examples of appropriate statements are:

- This thesis may be made available for loan and limited copying and communication in accordance with the Copyright Act 1968.
- This thesis may be made available for loan. Copying and communication of any part of this thesis is prohibited for two years from the date this statement was signed; after that time limited copying and communication is permitted in accordance with the Copyright Act 1968.
- This thesis is not to be made available for loan or copying for two years following the date this statement was signed. Following that time the thesis may be made available for loan and limited copying and communication in accordance with the Copyright Act 1968.
- (A statement of conditions applying to loan and access for copying which is consistent with any existing (intellectual property or other kind of) agreements relating to the thesis or work reported in it.)

Sarawat Anam
June 2016

Acknowledgements

First of all, praise is to Almighty Allah, The most Gracious and The most Merciful. Without His blessings, nothing can be achieved in my belief. I would like to express my deepest sense of gratitude to my supervisors Associate Professor Byeong Ho Kang, Dr. Yang Sok Kim and Dr. Qing Liu for their invaluable help, guidance, encouragement and patience during the course of my PhD. I am always grateful to Associate Professor Byeong Ho Kang and Dr. Yang Sok Kim for their kind help in programming, writing and feedback promptly. I would like to acknowledge to University of Tasmania (UTAS) and Commonwealth Scientific and Industrial Research Organization (CSIRO), Hobart for financially supporting my research. The Autonomous Systems, Digital Productivity and Service Flagship and the Tasmanian node of the Australian Centre for Broadband Innovation are assisted by a grant from the Tasmanian Government which is administered by the Tasmanian Department of Economic Development, Tourism and the Arts.

My respect and gratitude go to my parents and brothers for their continuous support and encouragement to pursue the higher study. I am especially grateful to my father for his encouragement to do a PhD. I would like to thank my beloved husband and son for their understanding, patience and support to my study. I would like to give especial thanks to my husband for his patience for taking care of our little son.

I would like to thank Graduate Research Coordinator, Dr Leonie Ellis for her encouragement and help. I am grateful to Dr. Erin Roehrer and Dr.James Montgomery for checking English grammar in the conference papers. I am also grateful to Dr. Louise Oxley, Dr. James Montgomery and David Herbert for checking English grammar in the thesis. I would also like to thank Dr. Ivan Bindoff, Dr. Robert Ollington and Dr. Saurabh Garg for their help. I thank my colleagues of Smart Service Systems lab for creating a good environment to do research in the lab. I would like to acknowledge the

support from the staff members of the University of Tasmania especially the staff in the School of Engineering and ICT.

I am grateful to all the staff members of CSIRO specially Dr Craig Lindley, Dr. Heiko Mueller and Dr. Daniel Smith for their help. Last of all, I would like to thank anonymous reviewers who reviewed my papers during my PhD study. All valuable comments from the reviewers helped to improve my research in this thesis.

Preface

This thesis presents research work done at Smart Service Systems Laboratory, School of Engineering and ICT, University of Tasmania. The research is supported with Tasmanian Graduate Research Scholarship (TGRS) and top-up scholarship from CSIRO Computational Informatics, Hobart. The thesis focuses on the schema mapping using knowledge-based schema matching system. The list of publications made out of the research is given in the following:

Conference Papers

1. Anam, S., Kim, Y. S., Kang, B. H., and Liu, Q. (2015). Designing a Knowledge-based Schema Matching System for Schema Mapping. In the Thirteenth Australasian Data Mining Conference (AusDM), in the Australian Computer Society series Conferences in Research and the Practice in Information Technology, volume 168, pages 68-77. ACM.

2. Anam, S., Kim, Y. S., Kang, B. H., and Liu, Q. (2016). Adapting a Knowledge-based Schema Matching System for Ontology Mapping. In the Thirty-Ninth Australasian Computer Science Conference (ACSC) 2016, in the Australian Computer Society series Conferences in Research and the Practice in Information Technology. ACM.

3. Anam, S., Kim, Y. S., Kang, B. H., and Liu, Q. (2015). Schema Mapping Using Hybrid Ripple-Down Rules. In the Thirty-Eighth Australasian Computer Science Conference (ACSC) 2015, in the Australian Computer Society series Conferences in Research and the Practice in Information Technology, volume 159, pages 17-26. CRPIT.

4. Anam, S., Kang, B. H., Kim, Y. S., and Liu, Q. (2015). Linked Data Provenance: State of the Art and Challenges. In the 3rd Australasian Web Conference (AWC), in the Australian Computer Society series Conferences in Research and the Practice in Information Technology, volume 166, pages 19-28. CRPIT.

5. Anam, S., Kim, Y. S., Kang, B. H., and Liu, Q. (2014). Evaluation of Terminological Schema Matching and Its Implications for Schema Mapping. In Pacific Rim International Conferences on Artificial Intelligence (PRICAI), LNAI 8862, pages 561-572. Springer International Publishing Switzerland.

6. Anam, S., Kim, Y. S., and Liu, Q. (2014). Incremental Schema Mapping. In the 2014 Pacific Rim Knowledge Acquisition Workshop (PKAW), LNCS 8863, pages 69-83. Springer International Publishing Switzerland.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This dissertation investigates the problem of schema mapping among heterogeneous schemas on the Web. I propose a new incremental Knowledge-based Schema Matching System (KSMS) to compute the semantic similarities between elements or attributes of two schemas, using several semantic similarity measures, linguistic thesauri resources, text processing techniques, a hybrid approach which is the combination of both machine learning and knowledge engineering approaches, and a graph matching algorithm.

In this chapter, I present the background and motivation of my research. In the background, I describe schema, heterogeneities of schemas, schema matching and mapping, and illustrate these using a simple example. Description of the heterogeneities of schemas in Section 1.1.1 demonstrates the importance and pervasiveness of schema mapping by showing it as a fundamental step in several database applications. Section 1.1.2 highlights the importance of using the most suitable matching approaches for schema mapping. I then present the research problems in Section 1.2. I scope the input, process and output information considered in my research in Section 1.3. I then describe the aims of this research in Section 1.4. Finally, I provide the thesis structure highlighting the contents of each chapter.

1.1 Background and Motivation

A *schema* is defined as a formal structure that represents a set of elements (Peukert et al., 2012). Each schema element has a name, a data type, a description (called

annotation) as well as instances. Schemas can be relational database schemas, XML schemas, entity-relationship diagram, and ontology description (Peukert et al., 2012). For different application domains, schemas can be available in many different formats and definition languages such as Structured Query Language (SQL) for relational database schemas, the Document Type Definition (DTD), XML Schema Definition (XSD) and XML Data Reduced (XDR) for XML schemas, Resource Description Framework (RDF), RDF schema (RDFS) and Web Ontology Language (OWL) for ontologies (Hai, 2005). The broad range of languages contributes to the pervasive use of schemas in data management and processing applications. An example of schemas is given in Figure 1.1:

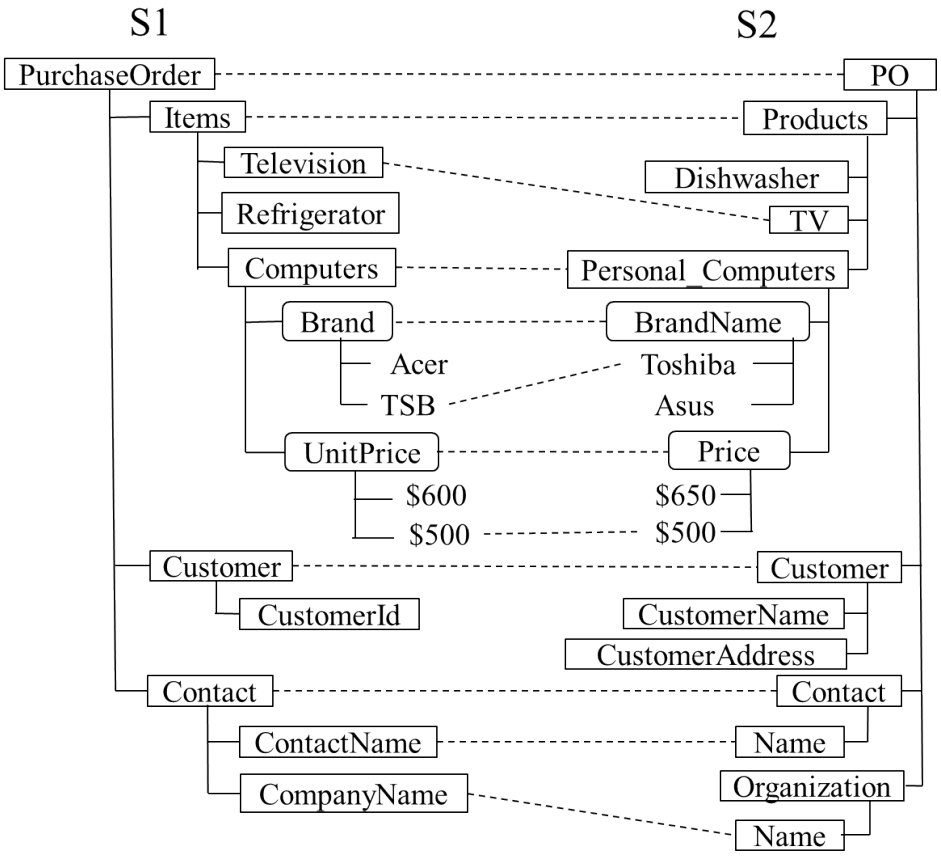

Figure 1.1: Example of two schemas

I use two XML schemas - a source schema ($S1$) and a target schema ($S2$) shown in Figure 1.1. Each schema consists of a set of schema elements. For instance, $S1$ consists of *PurchaseOrder*, *Items*, *Computers*, *Brand*, *UnitPrice*, *CustomerId*, *Contact*,

contactName and so on, and $S2$ consists of *PO*, *Products*, *Dishwasher*, *TV*, *Name*, *CustomerAddress*, *Organization* and so on. These schema elements may have data types such as string and integer. The elements of the schemas are in a hierarchical structure. *PurchaseOrder* and *PO* are roots of the schemas S1 and S2. *Items*, *Customer* and *Contact* are children of *PurchaseOrder*. *Television*, *Refrigerator* and *Computers* are children of *Items*. The children of *Computers* are *Brand* and *UnitPrice*. The schema elements may have some instances (data). For example, *Brand* and *UnitPrice* have instances *Acer* and $500 respectively. The children of *Contact* are *ContactName* and *CompanyName* in the $S1$ schema. The children of *PO* are *Products*, *Customer*, *Contact* and *Organization* in the $S2$ schema.

1.1.1 Schema Heterogeneity

A large number of schemas that describe the data structures of various domains are available on the Web. Examples of domains are purchase order, health, publication, geography, agriculture, environment and music. The schemas offer different kinds of element names, data types, allowable values, structures and grouping of elements and integrity constraints. The kind of information may vary between schemas as they are developed independently by different people for distinct goals in the real world (Hai, 2005). Examples of such variations are the following:

- Different possible representation models can be chosen for the schemas such as SQL, XSD, XDR, RDF and OWL (Aumueller et al., 2005; Cruz et al., 2009; Mao et al., 2010). These models are elaborated in Chapter 2.
- Distinct element names can be used to represent the same semantics (Hai, 2005). For example, *Items* is a synonym of *Products* in the purchase order domain. The same semantics can be represented by different structure. For example, *Computers* in $S1$ is represented as *Personal_Computers* in $S2$ in Figure 1.1. The same element name and structure can be used for different semantics. For example, if *name* is a child of both schema elements *Television* and *Dishwasher*, the semantics of *name* of both schema elements are different.
- Element names that are abbreviated may only be comprehensible to their creators (Hai, 2005; Cheatham and Hitzler, 2013b). For example, *PO* is an abbreviation

of *PurchaseOrder* in Figure 1.1. The names can be represented in combination of some stop words such as *is*, *to*, *beside* (Hai, 2005; Cheatham and Hitzler, 2013b). For example, *POShipTo* consists of *PO*, *Ship*, *To* where *To* is a stop word. The names can also consist of more than one word (Hai, 2005; Cheatham and Hitzler, 2013b). For example, *BrandName* consists of *Brand* and *Name*.

- Several data types such as string, varchar and integer, and integrity constraints such as unique, primary key and foreign key can be found in the same element names of separate schemas (Rahm and Bernstein, 2001; Hai, 2005).

- Element names can be represented at different levels of detail such as 1:1, 1:n, n:1, and n:m which are called matching cardinalities (Rahm and Bernstein, 2001). For example, *name* field of one schema can be captured by the combination of two fields $firstname$ and *lastname* in another schema which is called 1:n. The opposite of 1:n is n:1. More than one element of one schema can be captured by more than one element of another schema; it is then called n:m matching cardinality. *Contact* of $S1$ can be captured by *Contact* of $S2$ (see Figure 1.1) which is called 1:1. These matching cardinalities are described in more detail in Chapter 3.

- Different instances can represent the same semantics (Hai, 2005). For example, *TSB* in $S1$ and *Toshiba* in $S2$ (see Figure 1.1). Instance data may contain some errors such as misspellings, missing values and duplicate records (Hai, 2005).

- The hierarchical structures of schema element names may represent the same context of two schema graphs. In Figure 1.1, the hierarchical structure such as $PurchaseOrder.Contact.companyName \rightarrow PO.Organization.Name$ represents the same semantics. This is because *company* and *organization* are synonyms. So the child *Name* of *Organization* and *CompanyName* represent the same semantics. The hierarchical structures of schema element names can be represented in reverse order. For example, $PurchaseOrder.contactName \rightarrow Name.PO$.

Since the above variations are common in different schemas, managing semantic heterogeneities among them is an ongoing problem. The problem can be solved by *schema mapping* that is a high-level specification of the relationship between elements of two schemas called a source schema and a target schema (Cate et al., 2013). In

Figure 1.1, the dash lines indicate schema mappings or correspondences (Bernstein et al., 2011). Schema mapping is required in order to support comprehensive query and analysis capabilities in many application domains where data sharing between independently designed schemas is an issue. Some applications are the following:

- **Data integration:** Integrating data residing in different schemas.
- **Schema Evolution:** Migrating the deployed schemas into the new format.
- **E-commerce:** Translating the differently formatted messages of the trading partners.

The motivation for doing schema mapping among multiple heterogeneous sources comes from the importance of the above applications. I describe the applications and illustrate how semantic correspondences are required for the applications in Chapter 2. Schema mapping can be conducted by schema matching systems, which combine different matching algorithms with a mapping selection module (Ngo et al., 2013).

Schema Matching

Schema matching is a process that identifies semantic correspondences or mappings between related elements of schemas (Bellahsene et al., 2011). A mapping/correspondence is defined as a relationship between one or more elements of one schema and one or more elements of another schema (Rahm, 2011). Given two input schemas: $S1$ and $S2$, the match process returns as output mappings between certain elements of $S1$ and $S2$, also called the match results (Hai, 2005). The result of schema matching is between 0 (strong dissimilarity) to 1 (strong similarity) indicates the plausibility of the correspondence between schema elements (Do and Rahm, 2002). Each mapping may have a mapping expression which specifies how the schema elements of $S1$ and $S2$ are related to each other. According to Rahm and Bernstein (2001), the mapping expression may use simple relations over scalars (e.g., identity), terminological relationships (e.g., synonymy, hypernym, is-a, part-of), set-oriented relationships (e.g., equivalence, overlapping, subsumption), or functions (e.g., string concatenation or arithmetic functions).

Schema Matching Example

A schema matching example is given in Figure 1.1. In the figure, *Items* in $S1$ semantically corresponds to *Products* in $S2$ using a dictionary containing synonyms. *UnitPrice* in $S1$ semantically corresponds to *Price* in $S2$ because element name *price* in both schemas are the same, or the data types of both elements are the same. Another way of relating two schema elements is by looking at similarities in the pattern of their data values.

Schema Mismatch Examples

Schema mismatches can be found in distinct naming conventions, structures and hierarchical contexts. Some examples are described here. In Figure 1.1, $Refrigerator$ in $S1$ and *Dishwasher* in $S2$ do not semantically correspond to any of the schema elements of both schemas. The schema element *Customer* is modelled differently using another structure in both schemas. In schema $S1$, schema element *Customer* is modelled as *CustomerId*, and in schema $S2$, it is modelled as *CustomerName* and *CustomerAddress*. The schema elements do not semantically correspond. In the figure, the hierarchical structure such as $PurchaseOrder.Contact.contactName \rightarrow PO.Organization.Name$ does not represent the same semantic. This is because *Contact* and *Organization* do not match semantically.

1.1.2 Requirements for Schema Match Results

Similar to the heterogeneities in schemas, there is also variation in the results of schema matching because of distinct approaches used to predict correspondences. The main challenge is to conduct schema matching at different levels and to select the most suitable match approaches to execute for a given domain.

Schema matching can be conducted manually, semi-automatically and automatically. Manual matching is a time-consuming, error-prone and expensive process. In manual matching, it is not easy for a human expert to provide accurate matching information as the schema data changes over time. In addition, different experts may provide different opinions about the correctness of the mapping. It is therefore necessary to remove the burden of manual matching using semi-automatic and automatic match-

ing for large databases. However, fully-automated schema matching is not feasible because of the complexity of the schemas. For this, semi-automatic schema matching is required to improve the above two matching processes.

Schema matching can be performed at the element-level and structure-level. *Element−level matching* considers matching only names of elements. The matching process is described in Figure 1.2 (Shvaiko and Euzenat, 2005):

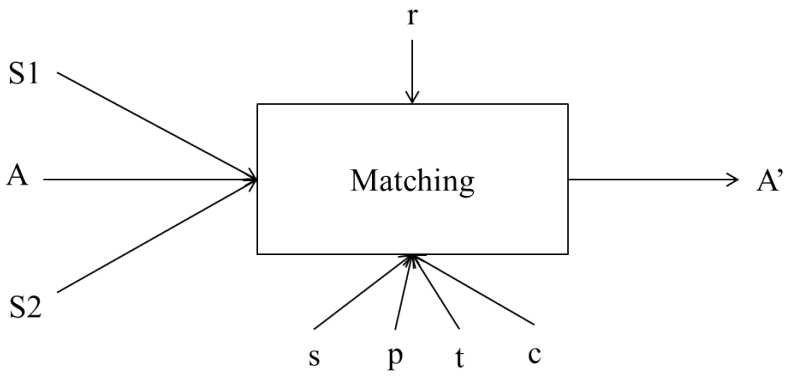

Figure 1.2: The schema matching process

In Figure 1.2, the matching process determines the alignment (A') for a pair of schemas ($S1$ and $S2$) where certain elements of $S1$ are mapped to certain elements of $S2$. A set of mapping elements is called an alignment (Shvaiko and Euzenat, 2005). The parameters of the matching process are: the input alignment (A) found from experts manual mapping needs to be completed by the matching process; the matching operations such as string similarity metrics (s) (Cohen et al., 2003), text processing techniques (p)(Cheatham and Hitzler, 2013b), a threshold (t) and combination (c) approaches; and external resources (r) such as domain dictionaries, thesauri, synonym tables and WordNet (Miller, 1995). *String similarity metrics* calculate the similarities ranging from 0 to 1 between two elements. Examples are Levenshtein, JaroWinkler, and MongeElkan. *Text processing techniques* are tokenization, synonym lookup, abbreviation expansion, stemming and stop word removal. These techniques are required to pre-process elements of schemas which contain different characteristics such as the frequency of combined, abbreviated, synonym, and stemmed words. A *threshold value* is used to compare the confidence measure between two schema elements produced by matching operations. According to the definitions, an example of a schema matching

process is described below:

A matching algorithm based on a string similarity metric *Levenshtein* produces a similarity value 0.4 between $ContactName$ of $S1$ and $Name$ of $S2$ in Figure 1.1. If the threshold value is 0.4 for determining correct mapping, the matching algorithm considers that all the pairs of elements with a similarity value is greater than or equal to 0.4 as correct mapping. Another matching algorithm matches $CompanyName$ in $S1$ and $Organization$ in $S2$ using the combination of tokenization and synonym look up. First, $CompanyName$ is tokenized as $\{Company, Name\}$ and $Company$ and $Organization$ are then matched according to the meaning of the elements using synonym lookup, and returns mapping decision correct to the user. The matching between TSB in $S1$ and $Toshiba$ in $S2$ is performed according to the meaning of the elements, which determines correct mapping. The *combination approaches* are used to combine the features constructed using schema elements, string similarity metrics, text processing techniques and expert mappings. *Structure level matching* uses the results of element-level matching for matching the hierarchical structure that represents different contexts of a shared element of a full graph. An example of structure-level matching is described in Section 1.1.1.

The *basic approaches* of element-level matching are string similarity metrics and text processing techniques, and these are termed terminological matching approaches. They compare matching schema elements. Different string similarity metrics and text processing techniques perform well for distinct schema elements. This is because the schema elements contain different characteristics. It is therefore necessary to combine these techniques effectively. The *Combination approaches* based on machine learning and knowledge engineering, have been proposed in the available literature (Ngo and Bellahsene, 2012; Aumueller et al., 2005) for schema mapping. These approaches are described below:

Machine Learning Approaches: Machine learning approaches are promising for predicting element similarity, but they require training datasets for building models. It is usually very difficult to obtain appropriate training datasets for large datasets and to change the trained models once mapped. Schemas can be created incessantly over time, and thus knowledge also changes over time. For handling new schemas, machine learning approaches need to rebuild their training models after collecting sufficient

data. Therefore, an approach is required that can easily update knowledge whenever new knowledge for mapping is found.

Knowledge Engineering Approaches: Knowledge engineering approaches can be applied where human heuristics are required to identify the mappings. These approaches are designed to perform a task where human experts exist, and also to assist human experts in making their decisions more consistent or of an improved quality. They are suitable for a small and well-defined domain. For analyzing a complex domain, they require time-consuming knowledge acquisition as in those systems highly trained specialists, a knowledge engineer, and a time-poor domain expert are required (Richards, 2009).

Based on the problems of the above machine learning and knowledge engineering approaches, it is necessary to build an approach that can manage schema mapping knowledge using an incremental knowledge acquisition process. As distinct approaches may predict different element mappings, it is still necessary to verify and validate the mappings.

1.2 Research Problems

The research problems of this thesis are the following:

1.2.1 Problem 1: Evaluation of the Terminological Matching Approaches

Terminological matching approaches are string similarity metrics and text processing techniques. Many string metrics have been suggested (Cohen et al., 2003). It is well known that the match performance of string metrics varies among different elements (Cheatham and Hitzler, 2013a). Similarity metrics produce numeric value ranging from 0 (strong dissimilarity) to 1 (strong similarity). It is therefore important to set a threshold value for deciding whether or not the source and the target schema elements are matched. Using only string similarity metrics do not provide a good performance for matching because of different characteristics of schema elements. For increasing perfor-

mance, string text processing techniques such as tokenization, synonym, abbreviation, stop word removal, stemming and translation are required. So the first research problem is to analyze the performance of different combinations of string similarity metrics and text processing techniques.

1.2.2 Problem 2: Developing a Hybrid Approach for Schema mapping

For managing knowledge of schema mapping that changes over time, I have chosen an incremental knowledge engineering approach, Censor Production Rules (CPR) based Ripple-Down Rules (RDR) (Kim et al., 2012) as it overcomes the 'knowledge acquisition bottleneck' problem of conventional knowledge engineering approaches, and maintains rules and simple knowledge acquisition. It allows the user to incrementally build the knowledge base while the system is in use, with no outside assistance or training from a knowledge engineer (Compton et al., 1991). However, the limitation of the approach is that it requires considerable time to create rules for mapping pairs of elements one by one. Though machine learning approaches and the CPR based RDR approach have some advantages, no single approach can manage schema mapping knowledge efficiently. In order to exploit the advantages and reduce the limitations of both machine learning approaches and the CPR based RDR approach, it is necessary to build a hybrid approach called Hybrid-RDR that can combine a machine learning approach and the CPR based RDR approach. The second research problem is therefore to develop a hybrid approach that can overcome the limitations of machine learning and knowledge engineering approaches.

1.2.3 Problem 3: Developing a Knowledge-based Schema Matching System (KSMS)

The Hybrid-RDR approach works for element-level matching. This considers matching names of element pairs, and alone does not give accurate results. Structure-level matching is therefore required to obtain accurate results. Structure-level matching that considers the hierarchical structure of the schema is required to adjust incorrect matches found from element-level matching. Based on the necessities, the third re-

search problem is to develop a KSMS that will perform element-level matching using the Hybrid-RDR approach and structure-level matching using the Similarity Flooding (Melnik et al., 2002) algorithm. This algorithm uses the concept that two nodes are similar when their neighbor elements are similar. The final mappings require to be generated by combining the results of element-level matching and structure-level matching using aggregation functions.

1.2.4 Problem 4: Developing a Simulated Expert-based Hybrid Approach for Schema mapping

In the Hybrid-RDR approach, it is necessary to populate the KB by creating rules by human experts to correct matches when machine learning approach gives incorrect matches. Sometimes a human expert may not be available to populate the KB by providing accurate information. In order to maintain the knowledge acquisition for schema mapping without human intervention, it is necessary to develop a Simulated Expert based Hybrid-RDR approach.

1.3 Scope of the Research

The match results consist of corresponding schema elements but does not exactly specify how the elements are related to each other without mapping expressions (Hai, 2005). The first step of creating a semantic mapping between two schemas is *schema matching*, which identifies corresponding elements (Hai, 2005). The second step is called *query discovery*. This enriches the identified correspondences with real mapping expressions in order to translate data instances of the mapped source schema into those of the target schema (Shvaiko and Euzenat, 2005). Most mapping research considers the first step for creating correspondences. Like previous work, I focus on the first step to identify correspondences between pairs of schema elements without mapping expressions. The scope of this research is as follows:

- Most mapping research has been carried out among database schemas, XML-schemas, and ontologies. My research focuses on schema mapping in XML-schemas on the Web and in ontologies on the Semantic Web. XML-schemas

have been widely used in many enterprises and organizations for describing their structured data.

- I consider schemas containing different types of characteristics such as the frequency of identical, abbreviated, synonym, stopped and combined words, and different structures. For example, in Figure 1.1, *Contact* of $S1$ is an identical word to *Contact* of $S2$. Examples of other features are given in Section 1.1.1.

- Most schema matching systems consider only 1:1 matching cardinality. This is because deriving complex matches is very difficult. A complex match is defined as a match with cardinalities 1:n, n:1 and n:m. Determining complex matches implies that every combination of schema elements of one schema is a potential match for every combination of schema elements of another schema. The potential number of complex matches is unbounded (Doan and Halevy, 2005). I do not consider complex matches in this research. I use only 1:1 matching cardinality that compares every element of the first schema to every element of the second schema (this means the evaluation of the Cartesian product or cross join) for determining matching.

- Matching can be performed at the instance or data-level and schema-level (Rahm and Bernstein, 2001). Instance-level matching is defined as data element matching between two schemas. For instance, matching between TSB of $S1$ and *Toshiba* of $S2$ in Figure 1.1. Schema-level matching is conducted between schema element pairs of a source schema and a target schema; for instance, matching between *Brand* of $S1$ and *BrandName* of $S2$ in Figure 1.1. Instance contains useful information, but it makes the solution space larger. For large schemas, considering instance data is very difficult. I therefore only consider schema-level matching.

- I do not use data types and integrity constraints in this research for reducing the complexity of schema mapping.

- I consider matching the hierarchical paths of schemas, but I exclude reverse paths. An example of a reverse path is given in Section 1.1.1.

- I use domain knowledge dictionaries created by COMA (Do and Rahm, 2002) for schema mapping, and a global knowledge thesaurus, WordNet (Miller, 1995), for

ontology mapping.

- In order to evaluate the match performance/quality of approaches for schema mapping, I use manual matches (TRUE/FALSE) between schema elements to produce expert mappings (Giunchiglia and Shvaiko, 2003). The manual matches are called gold standard (Do et al., 2003).

- Using one approach does not perform well; a combination of multiple matching approaches is therefore an efficient way for measuring similarities between element pairs of schemas (Rahm and Bernstein, 2001; Do and Rahm, 2002). The main goal of this research is to use the Hybrid-RDR approach that combines a machine learning approach and an incremental knowledge engineering approach.

1.4 Research Aims

The goal of schema mapping is to reduce manual effort as much as possible by a semi-automatic matching process. In order to solve the research problems described in Section 1.2, the research aims are the following:

1.4.1 Performance Checking of the Terminological Matching Approaches

In order to evaluate the terminological matching approaches, the research aim is to examine the following three questions:

- Which string similarity metric and which threshold value generate better accuracy for schema mapping?

- How do string text processing techinques increase the performance of string similarity metrics for schema mapping?

- How do different combinations of string similarity metrics and text processing techniques improve performance for schema mapping?

1.4.2 Feature Construction

Before using the combination approaches machine learning and knowledge engineering for schema mapping, the research aim is to construct features of schemas using schema elements, string similarity metrics, text processing techniques and expert mappings (TRUE/FALSE).

1.4.3 Evaluation of the Machine Learning and Knowledge Engineering Approaches

In order to use machine learning and knowledge engineering approaches, the research aim is to examine the following questions:

- Which machine learning approach performs best for schema mapping?
- How does a knowledge engineering approach, CPR based RDR, increases performance of schema mapping using an incremental knowledge acquisition process?

1.4.4 Proposing a Hybrid Approach

Machine learning approaches require to rebuild a training model if schema elements changes over time, and the CPR based RDR approach requires time to create rules one by one for schema matching. In order to overcome these limitations, the research aim is to propose a hybrid approach called Hybrid-RDR that combines the machine learning approach, decision tree - J48 and the CPR based RDR approach.

Developing Graphical User Interface (GUI) for the Hybrid Approach: A fully automatic schema matching system is not possible as it is hard to identify all matches automatically without user interaction. A user-friendly interface is therefore essential for the practicability and effectiveness of a system. The interface requires to be designed to set some parameters for correcting and validating the automatically derived match results. Only a few existing schema matching systems, COMA/COMA++ (Do and Rahm, 2002; Aumueller et al., 2005), YAM/YAM++ (Duchateau et al., 2009; Ngo and Bellahsene, 2012), SAMBO (Lambrix and Tan, 2006), AMC (Peukert et al.,

2011) and MaF (Martinez-Gil et al., 2012) support GUI. However, in these systems, users need to add false negatives (relevant matching among schemas is found to be irrelevant) and remove false positives (irrelevant matching among schemas is found to be relevant) (Marie and Gal, 2008) one by one by manually clicking on the relevant schema elements shown within the GUI. Manually solving incorrect matches require users' proper knowledge about the domain, and this process is time-consuming, error-prone and expensive. Therefore, the research aim is to build a robust GUI for the Hybrid-RDR approach which can allow users to validate and verify schema matching results by an incremental knowledge acquisition process.

1.4.5 Producing Final Mapping Results

Element-level matching is not sufficient to obtain accurate results, structure-level matching is therefore required to determine accurate mapping results. In order to produce the final mapping results, it is necessary to use aggregation functions that combine the results of both element-level matching and structure-level matching. So the research aim is to develop a KSM that can perform both element-level matching and structure-level matching and combines the results using an aggregation function to generate the final mapping results.

1.4.6 Proposing a Simulated Expert-based Hybrid Approach

In order to reduce human involvement in the knowledge acquisition process to correct matches, the research aim is to propose a Simulated Expert-based Hybrid-RDR for schema mapping. Like the Hybrid-RDR approach, this approach combines the J48 and CPR based RDR approach. The only difference between the Simulated Expert-based Hybrid-RDR and Hybrid-RDR approaches is that knowledge acquisition is performed by a simulated expert and a human expert respectively.

1.5 Thesis Organization

This thesis is organized as follows:

Chapter 2

I present basic definitions and applications in Chapter 2. Different types of schemas can be used for schema mapping, and schemas can be represented by various formats and languages. For this, I provide definitions of different types of schemas, schema formats and languages, and a comparison between schema and ontology. I also represent the application domains of schema mapping in order to illustrate how semantic correspondences are required for the applications.

Chapter 3

In Chapter 3, I review the existing literature. First, I highlight the classification criteria of schema and ontology matching according to the previous survey. I then describe the matching dimensions such as input, process and output. I review the classifications of schema and ontology matching approaches and identify some research gaps at the terminological matching and combination approaches. To fill the gaps, I propose an incremental hybrid approach. Following this, I review the state of the art schema and ontology matching systems and summarize the advantages and limitations of the systems. In order to overcome the limitations of the existing systems, I propose a KSMS. Finally I provide the summary of this chapter.

Chapter 4

Chapter 4 deals with the shortcomings of the terminological schema matching approaches, knowledge engineering approaches and machine learning approaches. I first present a rational for evaluation of terminological matching approaches. Following this, I propose improving the performance using combination approaches such as knowledge engineering and machine learning approaches. I then present the terminological matching, a knowledge engineering approach and machine learning approaches. I also describe the schema datasets and evaluation metrics used in the whole thesis. Later, I describe the evaluation approach, experimental procedure and evaluation results of the approaches. After this, the experimental results of schema mapping are represented. Finally, I show the limitations of the approaches and a means for overcoming the limitations.

Chapter 5

I describe the hybrid approach in Chapter 5. I also represent the GUI for Hybrid-RDR, and show how it works. Later, I show the architecture and functionalities of the KSMS that performs both element-level matching and structure-level matching. I then describe the advantages of the Hybrid-RDR approach. Later, I evaluate the performance of the KSMS for schema and ontology mapping both at the element-level and structure-level using real world datasets. I then show the incremental performance of the dynamic decision tree and compare the performance to KSMS. I also compare the performance of KSMS to some existing systems.

Chapters 6

In Chapter 6, I illustrate simulated expert-based schema mapping and its experimental results. First, I explain the limitations of using Hybrid-RDR for schema mapping. I then introduce the simulated expert for the Hybrid-RDR approach and provide the algorithm for the Simulated Expert-based Hybrid-RDR approach. I describe the datasets, experimental procedure, evaluation process, and KB of this approach, and evaluation results. I then illustrate the advantages of this approach over Hybrid-RDR approach operated by human experts or users. I also show structure-level matching results that uses element-level matching results produced by the simulated expert-based Hybrid-RDR approach. Following this, I provide the final mapping results.

Chapters 7

In Chapter 7, I describe the conclusion of this thesis that consists of contributions and future work.

Chapter 2

Definitions and Applications

Given two schemas: $S1$ and $S2$, a schema match operation finds the most plausible mappings/correspondences between elements of the schemas by overcoming the *schema heterogeneities* described in Chapter 1. In a match operation, the input schemas specify the elements to be matched. It is therefore required to examine some typical schemas and their elements. The need for schema mapping in various applications has led to the development of many schema matching approaches and systems to semi-automatically solve match problems. For this, it is worth describing the applications to investigate how schema mapping works. I describe different types of schema and explain schema languages in Section 2.1. I describe the applications of schema mapping in Section 2.2. Finally, I conclude in the last section.

2.1 Schemas

In Chapter 1, I explain that schemas can be relational database schemas, XML schemas, entity-relationship diagram, and ontology description. Depending on the application domains, schemas can be represented in different formats and languages. Examples are SQL for relational database schemas, XSD and XDR for XML schemas and RDF, RDFS and OWL for ontologies.

2.1.1 Relational Database Schemas

A relational database (Codd, 1970) is based on the relational model of data, and this is managed by the software system known as the relational database management system (RDBMS). A relational database schema is described in a formal language supported by DBMS[1]. An example of such a schema is given in Figure 2.1 (Hai, 2005).

```
CREATE TABLE PurchaseOrder (
PONo INT,
DeliverNo INT,
PRIMARY KEY (PONo),
FOREIGN KEY (DeliverNo),
REFERENCES DeliverTo (DeliverNo)
);

CREATE TABLE DeliverTo (
DeliverNo INT,
Street VARCHAR(100),
City VARCHAR(100),
Zip VARCHAR(100),
PRIMARY KEY (DeliverNo)
);
```

Figure 2.1: An example of a relational schema

A relational database schema consists of a set of tables, e.g., *PurchaseOrder* and *DeliverTo*. Each table comprises a set of columns, e.g., the table *DeliverTo* consists of the columns *DeliverNo*, *Street*, *City* and *Zip*. A column in one table may be specified as a foreign key pointing to a column in another table to capture referential constraints between different elements. Element instances are stored as records of column values within a corresponding table. Structured Query Language (SQL)[2] allows the definition of schemas for relational databases, to query and manipulate data stored in such schemas.

[1] https://en.wikipedia.org/wiki/Database_schema

[2] http://www.w3schools.com/sql/

2.1.2 XML Schemas

Extensible Markup Language (XML)[3] is a uniform exchange syntax, which is the basis of Web data as well as Web services. The XML data is represented as a tree structure, and the data are both human and machine readable. XML is used for sharing structured and semantically rich data.

XML schemas: XML Schema Definition (XSD) and XML Data Reduced (XDR) are used in the e-Business industry for describing the structures and contents of business documents. Examples of XSD and XDR are shown in Figures 2.2 and 2.3 respectively.

```
< xsd : element  name = PurchaseOrder  type = PurchaseOrderType >
< xsd : complexType   name = PurchaseOrderType >
< xsd : sequence >
< xsd : element  name = DeliverTo  type = USAddress/ >
< xsd : element  name = InvoiceTo  type = USAddress/ >
< /xsd : sequence >
< /xsd : complexType >

< xsd : complexType  name = USAddress >
< xsd : sequence >
< xsd : element  name = Street  type = xsd : string/ >
< xsd : element  name = City  type = xsd : string/ >
< xsd : element  name = Zip  type = xsd : decimal/ >
< /xsd : sequence >
< /xsd : complexType >
< /xsd : element >
```

Figure 2.2: An example of an XSD schema

The main components of an XSD schema are elements (e.g., *PurchaseOrder* and *DeliverTo*) and attributes (e.g., *USAddress*). The nested sub-elements (e.g., *Street*, *City*, and *Zip*) have atomic data types such as *string* and *decimal*. An XDR schema consists of ElementType and AttributeType.

[3] http://www.w3.org/TR/REC-xml/

```
< ElementType  name = "PurchaseOrder"  content = "eltOnly" >
        < element  type = "DeliverTo" / >
        < element  type = "InvoiceTo" / >
 < /ElementType >
< ElementType  name = "DeliverTo"  content = "empty" >
< AttributeType  name = "Street"  dt : type = "string" / >
< AttributeType  name = "City"  dt : type = "string" / >
< AttributeType  name = "Zip"  dt : type = "string" / >
< /ElementType >
```

Figure 2.3: An example of an XDR schema

2.1.3 RDF Schemas

Resource Description Framework (RDF)[4] is a directed, labeled graph data format for representing information in triple stores on the Web. Each triple contains three parts in the form of <s,p,o> where s, p, and o denote subject, predicate, and object respectively. An RDF is used to create structured data that form a data source D. D is typically represented by a directed labelled graph g. The edge e of the graph is directed from s to o and labelled with p. Each RDF triple can be expressed using a URI. The object can also be represented by a literal value. An example of RDF is given in Figure 2.4.

```
< rdf : RDF
xmlns : rdf = "http : //www.w3.org/1999/02/22 − rdf − syntax − ns#"
   xmlns : dc = "http : //purl.org/dc/elements/1.1/" >
   < rdf : Description  rdf : about = "http : //www.w3.org/" >
      < dc : title > World  Wide  Web  Consortium < /dc : title >
   < /rdf : Description >
< /rdf : RDF >
```

Figure 2.4: An example of RDF

In Figure 2.4, $http://www.w3.org$ is a subject, $http://purl.org/dc/elements/1.1/title$ is a predicate or schema and $World\ Wide\ Web\ Consortium$ is an object or instance. Here a predicate represents a relationship between a subject and an object.

[4] http://www.w3.org/RDF

RDF Schema (RDFS)[5] extends RDF vocabulary as a schema language for the RDF and a light-weight ontology language. It provides mechanisms for describing groups of related resources and relationships between these resources. These resources are used to determine the characteristics of other resources: domains and ranges of properties. The RDFS defines three types of entities: Class, ObjectProperty, and DatatypeProperty. Classes represent concepts in a domain. Object properties are relationships between classes. Datatype properties are relationships between classes and data types. In the RDFS, a certain object is an instance of a certain class. The RDFS can describe hierarchies between classes and properties for representing semantic relationships. RDFS documents are ontologies. An example of an RDFS is illustrated in Figure 2.5.

```
< rdf : RDF >
< rdf : Description rdf : ID = "PurchaseOrder" >
  < rdf : type rdf : resource = "http : //www.w3.org/2000/01/rdf − schema#Class" / >
< /rdf : Description >
< rdf : Description rdf : ID = "Item" >
  < rdf : type rdf : resource = "http : //www.w3.org/2000/01/rdf − schema#Class" / >
  < rdfs : subClassOf rdf : resource = "#PurchaseOrder" / >
< /rdf : Description >
< /rdf : RDF >
```

Figure 2.5: An example of RDFS

2.1.4 Web Ontology Language (OWL)

Web Ontology Language (OWL)[6] is a Semantic Web language. It represents knowledge about things, a group of things and relations between things. OWL ontologies are directed labeled graphs. These are designed for use by applications that require the content of information to be processed instead of presenting information to humans. OWL facilitates greater machine interpretability of Web content. An example of OWL is given in Figure 2.6 (Hai, 2005). OWL defines classes (e.g., *PurchaseOrder*, *Organization*, and *Address*), their relationships using subClassOf (e.g., *Agent*) and

[5] http://www.w3.org/TR/rdf-schema/

[6] http://www.w3.org/2001/sw/wiki/OWL

properties (e.g., *hasAddress* of *Address*, *Street* of *Address*). The value range of properties may be an atomic type (e.g., *String* for *Street*).

```
< owl : Class rdf : ID = PurchaseOrder/ >
< owl : ObjectProperty rdf : ID = deliverTo >
< rdfs : domain rdf : resource = #PurchaseOrder/ >
< rdfs : range rdf : resource = #Organization/ >
< /owl : ObjectProperty >
< owl : Class rdf : about = #Organization >
< rdfs : subClassOf rdf : resource = #Agent/ >
< /owl : Class >
< owl : Class rdf : ID = Address/ >
< owl : ObjectProperty rdf : ID = hasAddress >
< rdfs : domain rdf : resource = #Agent/ >
< rdfs : range rdf : resource = #Address/ >
< /owl : ObjectProperty >
< owl : DatatypeProperty rdf : ID = Street >
< rdfs : domain rdf : resource = #Address/ >
< rdfs : range rdf : resource = #String/ >
< /owl : DatatypeProperty >
< /owl : Class >
```

Figure 2.6: An example of OWL

In general, a schema is defined as a set of schema elements that are connected by structures. In relational database schemas, schema elements consist of names of tables and columns, and schema structure includes relationships between tables and columns and referential constraints expressed by foreign keys (Hai, 2005). In an XSD schema, XML elements and attributes are used as schema elements, and relationships between elements and sub-elements described by complex types are used as schema structure. In an XDR schema, schema elements consist of names of element types and attribute types, and schema structure includes the hierarchical structure defined between schema elements. In OWL ontology, schema elements are defined by classes, object properties and data type properties, and schema structure consists of relationships between classes and properties.

Comparison between Database Schema and Ontology

There are some differences and similarities between a database schema and ontology. Database schemas often do not provide explicit semantics. This is because semantics are specified at design time and are not available (Noy and Klein, 2004). However, ontologies are logical systems that provide explicit semantics, and these semantics are encoded in different ontologies (Shvaiko and Euzenat, 2005). The similarities between schemas and ontologies are that they both provide a vocabulary of terms in which domains of interest are described, and they both constrain the meaning of terms that are used in the vocabulary (Uschold and Gruninger, 2004). For matching ontologies semantically, domain ontologies, external dictionaries, and WordNet (Miller, 1995) are required. However, external dictionaries are not required for matching schemas because the meaning of schema elements is encoded in the schemas (Shvaiko and Euzenat, 2005). One key difference is that an ontology has a more complex structure than a schema, and an ontology can express class hierarchy through built-in property $rdfs : subClassOf$. Schemas allow for a tree-based definition of elements and types whereas ontologies are graph structured.

2.2 Application Domains

The goal of schema mapping is to reduce manual effort as much as possible by a semi-automatic matching process. For an example, a user wants to buy a computer under 500 dollars. He starts to search Web-based databases of different shops that sell electronic products. A Web service application provides services for searching computers in the databases of different retailers based on users' input parameters. The problem is that all of the databases have been created independently, and the same product may have different names. For example, "*select computers from database where brand is 'Toshiba' and UnitPrice is $500"*. The query will not work for all databases. For making the query workable, schema mapping is required. After determining mappings between pairs of elements of two schemas, it is necessary to generate query expressions to translate data instances of the mapped schema elements under an integrated schema (Shvaiko and Euzenat, 2005). If users provide the query to the integrated database, it will then work to retrieve the appropriate answer.

Schema/ontology mapping is necessary in many applications such as data integration, data exchange, schema evolution and e-business (Glavic et al., 2010; Shvaiko and Euzenat, 2005). The applications are described below:

2.2.1 Data Integration

Data integration involves combining data residing at heterogeneous sources and providing users with a unified view of these data (Lenzerini, 2002). In data integration, schema mapping is used to translate queries from a source schema into a global schema from heterogeneous data sources. A typical example of data integration system according to Lenzerini (2002) is given in Figure 2.7.

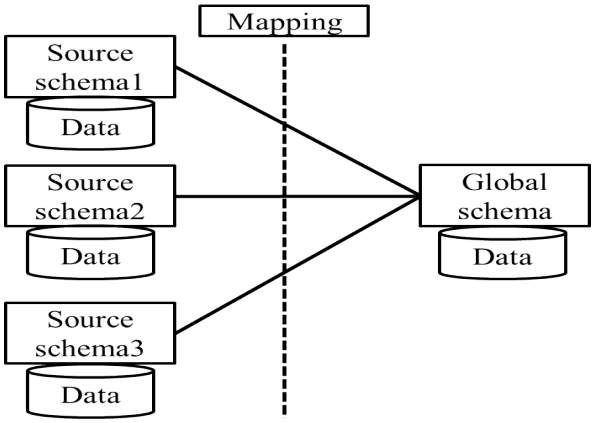

Figure 2.7: A data integration system

Data integration is performed incrementally by starting with a global schema and adding new schemas when needed. As schemas are developed by different people in various domains such as purchase order, education, music and agriculture, the schemas are different structurally and terminologically. For data integration of these schemas, two steps are required. In the first step, schema matching is performed to create mappings/correspondences between semantically similar pairs of elements of the schemas. In the second step, mapped elements are used to generate queries for transforming their instances from the source schema into the global schema. Some systems use schema mappings for data integration. Among them, the system developed by McBrien and Poulovassilis (2003) uses $both-as-view$ (BAV) for data integration in the relational data model. In the BAV approach, the definition of a global schema is achievable as

views over a local source schema (also known as global-as-view (GAV)) and the definition of a local schema is achievable as views over a global schema (also known as local-as-view (LAV)). The BAV approach supports the evolution of global and local schemas. This system (McBrien and Poulovassilis, 2003) is called bi-directional because it uses the BAV approach and bi-directional transformation rules to integrate a source schema into a target schema.

Another system (Dos Santos Mello and Heuser, 2005) uses a semi-automatic integration process for XML schema called BInXS, to provide a unified view of a lot of heterogeneous XML schemas on the Web. BInXS adopts the global-as-view integration approach that maps all elements and attributes definitions of XML schemas using the XPath language that maintains mappings among global concepts and data. During mapping, information of original schemas is to be preserved to retrieve the source XML schema when needed. Information preservation is needed for data integration (Lenzerini, 2002) and peer-to-peer database systems (Arenas and Libkin, 2005). For performing information preservation, two criteria: invertibility and query preservation (Bohannon et al., 2005) are used. Invertibility means one can retrieve a source schema from a target schema. Query preservation means the queries used for querying a source schema for a particular XML query language must be used for a target schema. The system (Bohannon et al., 2005) operates schema embedding for XML DTD schemas from which an instance-level XML mapping is automatically derived. Three types of problems: information preservation, type checking, and accommodation of multiple source schemas are therefore to be solved by using only simple navigational queries.

2.2.2 Data Exchange

During data exchange, schema mappings generate transformations that produce target instances into source instances. Mecca et al. (2009) explained that given a mapping scenario, it is possible to compute the core solutions for the data exchange problem. They also explained that mappings are executable transformations that specify how source instances can be translated into target instances by using SQL or XQuery scripts. Arenas and Libkin (2005) described some problems of XML data exchange such as source-to-target constraints, data exchange settings, consistency and query answering where XML schema mapping is required. Here they use navigational queries as well

as simple tree patterns binding several attribute values. Later, Amano et al. (2009) developed a language that considers three types of problems: static analysis of mappings, their complexity, and their composition of XML schema mappings. The static analysis problem is described based on consistency such as whether it is possible to map some elements of a source schema onto an element of a target schema or whether all elements of a source schema can be mapped. The FlexMap framework (Ramanath et al., 2003) for XML-to-relational mapping incorporates sets of schema transformations. In this framework, the impact of schema transformations and query workload on search algorithms and optimizations to speed up the algorithms is also defined. The mapping between XML schemas and relational databases and vice versa must be described in such a way that the structure and content can be preserved and retrieved when needed. The system developed by Barbosa et al. (2004) defines lossless mapping schemes. It preserves the structures and contents of the schemas. It also validates mapping schemas so that valid schemas can be mapped onto legal databases.

2.2.3 Schema Evolution

Schema evolution is defined as the ability to change deployed schemas such as relational databases schemas and XML schemas into the new format. Schema evolution is necessary for managing new requirements for propagating schema changes to instances, correcting deficiencies in the existing schemas, and migrating to a new platform (Hartung et al., 2011). In schema evolution, schema mapping involves the discovery of a query or a set of queries that transform a source schema to a new structure. In heterogeneous data sources, schemas evolve over time, and new data added to the exiting schema needs a new schema structure. When schemas change, mappings between these schemas may become invalid. It is necessary to facilitate adaptation and reuse the existing schema mappings. Yu and Popa (2005) developed a mapping adaptation system that combines mapping composition and mapping pruning techniques for adapting schema mappings to reflect schema evolution. Beyer et al. (2005) described a taxonomy of changes for XML schema evolution. They discuss the impact of schema evolution on schema validation and query evaluation. They also introduce a means of controlling schema changes, and write queries across schema versions for managing XML schema evolution.

2.2.4 E-Business

With the availability of the internet, trading partners can handle business transactions such as exchanging product information, placing purchase orders, and confirming and paying orders by exchanging electronic documents or messages. The contents of the messages are expressed in knowledge representation languages that are referred to some ontologies (Shvaiko and Euzenat, 2005). Ontologies that are designed independently may vary because of different names, data types, ranges of values, grouping of fields and languages. It is therefore necessary to match the ontologies for translating messages so that trading partners can understand the messages. The matching of ontologies establishes the semantic mappings between concepts called ontology alignment. As a consequence, trading partners exchanging their messages in different ontologies would be able to negotiate the matching of concepts in their respective ontologies and to translate the contents of the messages they exchange with the help of the mappings.

2.3 Summary of this Chapter

In this chapter, I described some definitions of different schema languages: relational, XSD, XDR, RDFs and OWL. I also described some applications of schema and ontology mapping: data integration, data exchange, schema evolution and e-business.

Chapter 3

Literature Review

As explained in the previous chapter, schema mapping is used in data integration, data exchange, schema evolution, and E-business. For making the schema matching tasks as automatic as possible, there has been considerable work in different fields such as Artificial Intelligence, databases and knowledge representation. The main goal of this chapter is to survey existing schema matching approaches and explain their common features, applicability, advantages and limitations in order to fulfill the research aims described in Chapter 1. This chapter is divided into the subject areas of classification criteria, matching dimensions, classifications of the schema and ontology matching approaches, and state of the art schema and ontology matching systems.

I review the previous surveys on schema and ontology mapping, and highlight the classification criteria according to the previous survey in Section 3.1. I explain the matching dimensions such as input, process and output in Section 3.2. I review the classifications of schema and ontology matching approaches in Section 3.3. While reviewing the classification approaches, I find that the basic approaches for element-level matching are terminological matching approaches, and the combined techniques are combination approaches. I identify some research gaps in the terminological matching and combination approaches, and these gaps are described in Section 3.3.1. In order to fulfill the gaps, I propose to build an incremental hybrid approach. In Section 3.4, I review the state of the art schema and ontology matching systems and summarize the systems. While reviewing the systems, I find some limitations. In order to overcome the limitations, I propose to build an incremental Knowledge-based Schema Matching System (KSMS). The last section contains a conclusion of this chapter.

3.1 Classification Criteria

Many diverse solutions have been proposed in the available literature for schema and ontology mapping. Rahm and Bernstein (2001) classified database schema matching approaches into two: individual matchers and combining matchers. Individual matchers consist of schema-based, instance-based, element-level, structure-level, linguistic-based and constraint-based. Combining matchers comprise hybrid matchers and composite matchers. The authors also presented a comparative review of the matching systems. According to them, the classifications of schema matching approaches are described in Figure 3.1:

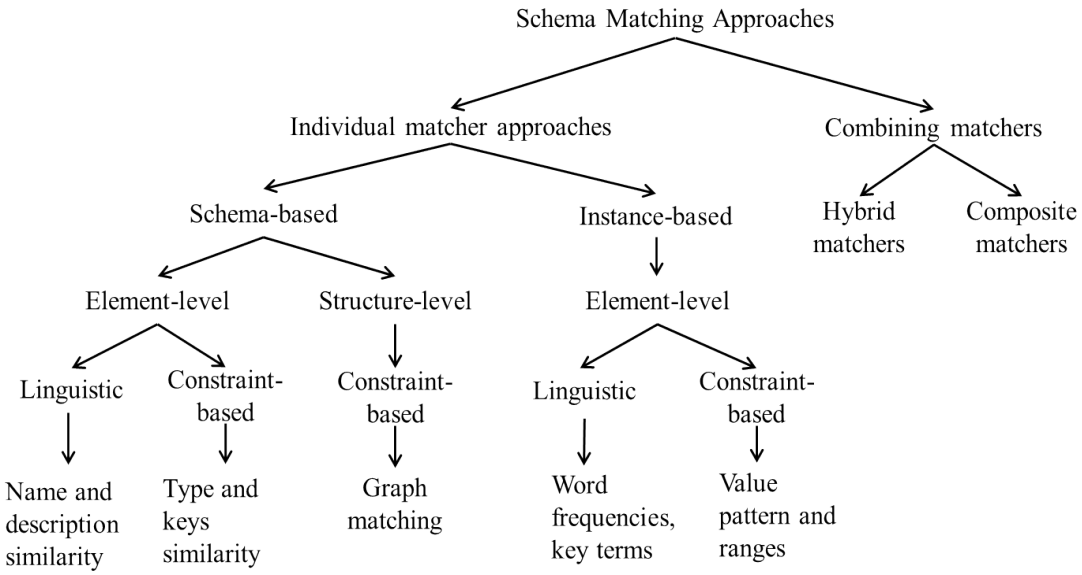

Figure 3.1: Classifications of schema matching approaches (Rahm and Bernstein, 2001)

Instance−based matching is defined as the matching between data elements (data contents). *Schema−based matching* considers matching between schema elements, e.g., element names, data types and structural properties. *Element−level matching* finds matching between elements by considering names individually and ignoring the structural relationships between them. At this level, element names, descriptions, keys and data types are considered as criteria. The matching results of this level are used for *structure−level matching* to identify the structural similarity of schema elements by analyzing the positions of the elements on the hierarchical structure of graphs. At the structure-level, name path, children and leaves are considered as criteria. *Linguistic*

matchers use external resources such as a dictionary or a thesaurus for matching elements based on linguistic relations (e.g. synonyms, hypernyms) between them. The *Constraint−based approach* considers data types, cardinalities and keys of schema elements. In *hybrid matchers*, different match criteria or properties (names and data types) are used within a single algorithm for matching elements. Hybrid matchers can combine several matching approaches to determine the match candidates. They provide better match performance compared to the separate execution of multiple matchers (Rahm and Bernstein, 2001). *Composite matchers* combine the results of individual matchers or hybrid-matchers. Matchers depend on input schemas and instances as well as some auxiliary information such as dictionaries, global schemas, previous match results and user input. Matchers provide the flexibility for selecting the match algorithms to be executed based on the matching task at hand. Moreover, the approach can combine individual match results. In a match result, one or more elements of a source schema may be aligned with one or more elements of a target schema, resulting in different cardinalities (e.g., 1:1, 1:n, n:1 and n:m) (Rahm and Bernstein, 2001).

A survey by Wache et al. (2001) focuses on the ontology-based approaches used for information integration. Shvaiko and Euzenat (2005) distinguished approximate and exact techniques at the schema-level, and syntactic, semantic, and external techniques at the element-level and structure-level. They also focused on the current state of the art schema and ontology matching approaches and systems, and provided a comparative survey of the existing schema and ontology matching approaches and systems. Ontology-based information extraction techniques were reviewed by Wimalasuriya and Dou (2010). Shvaiko and Euzenat (2013) reviewed the state of the art ontology matching systems and compared the performance of the recent systems. They also addressed some challenges for ontology matching.

3.2 Matching Dimensions

Shvaiko and Euzenat (2005) classified the schema matching systems according to three dimensions: inputs of the systems, characteristics of the matching process, and outputs of the systems. The dimensions are described below:

3.2.1 Input Dimensions.

The schema and ontology matching systems have been classified depending on the data/conceptual models they take as input. Examples of these systems are described below:

- mSeer (Chai et al., 2008), SEMINT (Li and Clifton, 2000), Automatch (Berlin and Motro, 2002), iMap (Dhamankar et al., 2004), eTuner (Lee et al., 2007) support the relational data model;
- Self Configuring system (Peukert et al., 2012), LSD (Doan et al., 2001), SMB (Marie and Gal, 2008), MatchPlanner (Duchateau et al., 2008) and YAM (Duchateau et al., 2009) support XML schemas;
- Cupid (Madhavan et al., 2001) and COMA (Do and Rahm, 2002) support both relational and XML data models;
- SKAT (Mitra et al., 1999) takes XML, IDL and text as input, TranScm (Milo and Zohar, 1998) input DTD, OO, relational, HTML and SGML data models;
- Some ontology matching systems such as GLUE (Doan et al., 2002), SAMBO (Lambrix and Tan, 2006), LogMap (Jimenez-Ruiz and Grau, 2011), Malform-SVN (Ichise, 2008), MaF (Martinez-Gil et al., 2012), RiMOM (Li et al., 2009), ASMOV (Jean-Mary et al., 2009), OMReasoner (Shen et al., 2014), XMap++ (Djeddi and Khadir, 2014), AOTL (Khiat and Benaissa, 2014) and MassMtch (Schadd and Roos, 2014) take OWL as input.
- Some other systems such as COMA++ (Aumueller et al., 2005), Anchor-Flood (Seddiqui and Aono, 2009), GOMMA (Kirsten et al., 2011), YAM++ (Ngo and Bellahsene, 2012), AgreementMaker (Cruz et al., 2009), Falcon (Hu et al., 2008) and PRIOR++ (Mao et al., 2010) support both RDFS and OWL data models.

The systems have also been classified according to the input information: instance-level and schema-level information. They are described below:

Instance-based Matching

Instance-based schema matching identifies the similarity between schema elements according to the similarity of instances associated with the schema elements (Rahm,

2011). The semantics of schema elements depend on the instances, so instance-based matching determines high-quality correspondences ((Rahm, 2011). Instance-based schema matching has the following advantages (Thor et al., 2007): 1) the number of instances is higher than the number of schema elements which helps to determine the degree of element similarity; and 2) the match accuracy of the approach can be high though there are some instance mismatches.

In order to determine the similarity between instances, all the instances of a schema element are combined into a virtual document. In this way, many virtual documents are created from all the schema elements. The documents are then compared with each other using the document similarity measure, TF/IDF, for completing matching. This approach has been implemented in some systems including RiMOM (Li et al., 2009) and COMA++ (Massmann and Rahm, 2008). COMA++ uses website names and descriptions for determining the similarity between documents. Instance overlapping methods are also used to determine elements similarity. In COMA++, URLs are used to identify the overlapping between web directories such as Yahoo and Google considering URL usage. Four similarity measures: base-k similarity, dice, minimum and maximum are used to determine URL-based similarity. URL matching alone achieves average F-measures 60% and 79% with the combinations of name and description matching respectively. Instance overlapping methods are also used to match large life science ontologies (Kirsten et al., 2007) and product catalogs (Thor et al., 2007). In order to match product catalogs, the similarity between associated instances is used for deriving the similarity between elements. The hyperlink between data sources and general object matching are also used for performing instance matching. Hoshiai et al. (2004) compared feature vectors between a pair of elements using keywords found in the instances, and the similarity between feature vectors are determined by a structural matcher. There are other instance-based matching systems such as GLUE (Doan et al., 2002), SAMBO (Lambrix and Tan, 2006) and SEMINT (Li and Clifton, 2000).

Instance-based matching is not feasible in the process of making knowledge discovery easy and systematic. This is because instance-based matching has some problems such as lack of expressivity, schema heterogeneity, entity disambiguation, and ranking of results (Jain et al., 2010b). These problems can be solved by schema-based matching between different schemas.

Schema-based Matching

Schema-based matching determines the similarity between schema elements. The available schema elements include different properties such as names, descriptions, data types and constraints, and relationships between the elements such as constraints, is-a/part-of, or containment relationships. The advantage of schema-based matching is that it benefits both the Artificial Intelligence and Semantic Web Communities for some applications such as querying, reasoning, data integration, data mining and knowledge discovery (Jain et al., 2010b). After determining mappings between two schemas, next step is to generate query expressions that automatically translate data instances of these schemas under an integrated schema (Shvaiko and Euzenat, 2005).

Some schema-based matching systems are S-Match (Giunchiglia and Shvaiko, 2003), Anchor-flood (Seddiqui and Aono, 2009), ASMOV (Jean-Mary et al., 2009) and Falcon (Hu et al., 2008). S-Match implements semantic matching by a decider propositional satisfiability. Anchor-Flood starts matching from a small number of concepts called anchor. It then incrementally matches neighbors such as super-concepts, sub-concepts, and siblings of each anchor until no further matches are found, thereby building small segments (fragments) out of the ontologies to be matched. The system uses terminological matchers, structural matchers and background knowledge base (WordNet). ASMOV (Jean-Mary et al., 2009) uses terminological matchers, structural and extensional matchers for similarity calculation between two ontologies, and examines disjoint-subsumption contradiction and subsumption incompleteness for semantic verification. The matching process is repeated with the obtained alignments input until no new correspondence is found. Falcon (Hu et al., 2008) is an automatic partition-based system that uses a divide and conquer approach for ontology matching.

Instance and Schema-based Matching

There are some schema and ontology matching systems that support both instance and schema-based matching. Among them, COMA++ (Aumueller et al., 2005) is a generic schema and ontology matching system where different types of schema information such as name, type, path, children and leaves, and auxiliary information such as user-defined synonym and abbreviation tables and previous match results are used. Schemas are internally encoded as DAGs (Directed Acyclic Graphs) and are analyzed using

string matching algorithms. The system provides an extensible library of matching algorithms. However, in this system, users need to select the best combinations of matchers, which is not easy in a fully unknown schema matching problem. For this, users depend on the default configuration of the system. But the configuration is not always appropriate to handle largely differing matching problems of diverse domains. Martinez-Gil et al. (2012) developed an ontology matching framework, MaF, that uses the largest number of algorithms, and this framework can test the largest combinations of algorithms. However, the framework does not have the scope to correct and validate the result if the selected algorithm produces incorrect matching results. Some other instance and schema-based systems are GOMMA (Kirsten et al., 2011), mSeer (Chai et al., 2008), eTuner (Lee et al., 2007), and YAM++ (Ngo and Bellahsene, 2012).

Usage-based Matching

Usage-based schema matching has been described in the available literature. Elmeleegy et al. (2008) proposed a usage-based schema matching approach that exploits information extracted from the query logs to find correspondences between attributes in the relational schemas to be matched. The approach does not depend on the schema information or the data instances. In the approach, co-occurrence patterns are first determined between attributes and additional features, such as in joins and with aggregation functions. A genetic algorithm is then applied to find the highest-score mappings according to the scoring function used to measure similarities between the features of the matching attributes. The advantage of this algorithm is that it can match schemas even if their attribute names are opaque, or the layouts of the schemas are different. Nandi and Bernstein (2009) developed a hamster approach that uses the click log for keyword queries of an element search engine. The approach determines matchings between schema elements if the distributions of keyword queries causing click-through on their instances are similar. However, the problem of usage-based matching is that it is very difficult to obtain suitable usage data (Rahm, 2011).

3.2.2 Process Dimensions

The classification of the matching process depends on the approximate or exact nature of its computation (Shvaiko and Euzenat, 2005). The absolute solution to a problem is

computed by the exact algorithms, and the approximate algorithms sacrifice exactness to performance (Ehrig and Sure, 2004). The techniques used for schema or ontology matching systems are either approximate or exact. Interpretation of the input data is another way to analyze the matching algorithms. Shvaiko and Euzenat (2005) proposed three large classes: syntactic, external and semantic at the element-level and structure-level based on the input, external resources and semantics of the considered schema elements.

3.2.3 Output Dimensions

The schema and ontology matching systems have been classified based on the match results they produce. An alignment is defined as a set of mapping elements where certain elements of a schema S1 are mapped to certain elements of a schema S2 (Shvaiko and Euzenat, 2005). Each mapping element of the match result can have a mapping expression that specifies how a source schema and a target schema are related. In mapping expressions, matching cardinality problems are also considered (Rahm and Bernstein, 2001). There are four types of matching cardinality problems: 1:1, 1:n, n:1 and n:m. Examples of matching cardinalities are described in Table 3.1 (Rahm and Bernstein, 2001).

Table 3.1: Examples of matching cardinalities

Cardinality	**S1 Element**	**S2 Element**	**Mapping Expression**
1:1	id	sid	id=sid
n:1	firstName	lastName	concat(firstName,lastName)= fullName
1:n	address	street, city	split(address)=$\{street, city\}$
m:n	N.StuName, N.DeptNo, D.DeptNo, D.DeptName	C.Student, C.Department	select N.StuName, D.DeptName from N, D where N.DeptNo=D.DeptNo = $\{C.Student, C.Department\}$

Table 3.1 illustrates the four local cardinality cases using four individual correspondences. If matching is done element by element meaning one element of a source

schema is matched with one element of a target schema, then it is called 1:1 matching cardinality. For example, *id* of a source schema S1 is matched to *sid* of a target schema S2. This mapping expression is represented by an equivalent correspondence. When two or more elements of a source schema are matched with one element of a target schema, it is then called n:1 matching cardinality. The matching of $firstname$ and *lastname* to $fullname$ is defined as n:1 matching cardinality. Matching cardinality, 1:n is opposite of n:1. If *address* is matched to *street* and *city*, then it is called 1:n matching cardinality. These n:1 and 1:n matching cardinalities are described by concatenation and split functions respectively in the mapping expression (see Table 3.1). When two or more elements of a source schema are matched with two or more elements of a target schema, it is then called n:m matching cardinality. An example of this matching cardinality is shown in the fourth row of the table which uses an SQL expression combining attributes from two source tables N and D in S1 for student and department data respectively to populate the target table C in S2. Examples of some schema and ontology matching systems which produce outputs using different matching cardinalities are described below:

- mSeer (Chai et al., 2008), eTuner (Lee et al., 2007), SMB (Marie and Gal, 2008), GLUE (Doan et al., 2002), SAMBO (Lambrix and Tan, 2006), LogMap (Jimenez-Ruiz and Grau, 2011), Malform-SVN (Ichise, 2008), MaF, (Martinez-Gil et al., 2012), RiMOM (Li et al., 2009), COMA++ (Aumueller et al., 2005), Anchor-Flood (Seddiqui and Aono, 2009), GOMMA (Kirsten et al., 2011), YAM++ (Ngo and Bellahsene, 2012), Falcon (Hu et al., 2008) and PRIOR++ (Mao et al., 2010) produce 1:1 match cardinality as output.

- ASMOV (Jean-Mary et al., 2009) generates only n:m match cardinality as output.

- Other systems such as AgreementMaker (Cruz et al., 2009), Embley et al. (2004) and Kirsten et al. (2007) generate 1:1, 1:n, n:1 and n:m match cardinalities as outputs.

3.3 Classification of the Schema and Ontology Matching Approaches

Rahm and Bernstein (2001) classified schema-based matching into element-level and structure-level, and instance-based matching into only element-level described in Figure 3.1. The schema and ontology matching approaches are described below:

3.3.1 Element-level Approaches

Element-level matching considers matching properties of schema elements using the linguistic-based approaches and constraint-based approaches described in Figure 3.1 to determine the correspondences between schema elements.

i. Linguistic-based Approaches

Linguistic-based approaches consider text-based properties of schema elements such as name and description. In particular, element names are used for assessing element similarities as they are the most basic constituent of schemas. Name similarity can be performed by the terminological matching approaches and combination approaches. The approaches are described below:

A. Terminological Matching

Terminological matching is a basic approach that compares matching elements using string similarity metrics and text processing techniques. String similarity metrics calculate the degree of similarity between elements. These metrics produce numeric values ranging from 0 to 1. Cohen et al. (2003) classified string similarity metrics into edit-distance like functions, token-based distance functions, and hybrid distance functions. I use these string metrics in my research. These are described below:

Edit-distance like functions: Edit-distance like functions match two strings, s and t, based on edit operations such as character insertion, deletion and substitution, and each operation must be assigned a cost. Edit-distance like functions are the following:

- **Levenshtein distance:** For calculating the similarity between two strings, Levenshtein distance considers the operations such as insertion, deletion and substitution for matching two strings $s1$ and $s2$. The edit distance is represented by $xform(s1, s2)$. For calculating the worst case transformation cost $xform\_wc(s1, s2)$, all parts of $s1$ are substituted with the parts of $s2$, then the remaining parts of $s1$ are deleted and the additional parts of $s2$ are added. String similarity is then computed.

- **NeedlemanWunsch:** NeedlemanWunsch distance gives a higher cost for insertion and deletion.

- **JaroMeasure:** JaroMeasure calculates a similarity between two strings by counting the common letters appearing at the same and different positions in these strings. JaroMeasure is calculated by: $Jaro(s1, s2) = 1/3(m/|s1| + m/|s2| + (m - t)/m)$, where the number of matching characters and the number of transpositions are represented by m and t respectively.

- **JaroWinkler:** JaroWinkler distance is a variation of the Jaro distance metric. It works better for longer prefixes and short strings. The formula of JaroWinkler is defined by Cheatham and Hitzler (2013b): $JaroWinkler(s1, s2) = Jaro(s1, s2) + (lp(1 - Jaro(s1, s2)))$, where l denotes the length of the common prefixes and maximum length can be 4, and p is a weight that considers the common prefix.

- **SmithWaterman:** SmithWaterman algorithm compares segments of all possible lengths for determining similar regions between two strings and optimizes the similarity measure.

- **Q-gram:** Q-Gram counts the number of tri-grams between two strings for matching.

Token-based Distance Functions: Token-based distance functions consider two strings s and t as multi-sets (or bags) of words (or tokens). These functions calculate the words in these two multi-sets. In this research, the following token-based string metrics are considered:

- **Jaccard:** Jaccard similarity coefficient compares two strings as sequences of tokens instead of sequences of characters. The Jaccard similarity between the

word sets S and T is defined by (|S (∪) T|)/(|S (∩) T|), where the unique words of the two strings are represented by the union of S and T and the common words are represented by the intersection of S and T.

- **TFIDF:** TFIDF stands for Term Frequency - Inverse Document Frequency. It is a technique used for document indexing in information retrieval systems. The term frequency is the number of times a word appears in a document divided by the number of words in the document. The inverse document frequency is the logarithm of the number of documents divided by the number of documents that contain the word in question.

- **Cosine:** Cosine calculates the similarity between two vectors that measure the cosine of the angle between them.

Hybrid distance functions: Hybrid distance functions use recursive matching scheme for comparing two long strings s and t. An example of this function is MongeElkan. MongeElkan compares two strings based on an internal character-based similarity measure (e.g. edit distance) combined with a token level (i.e. word level) similarity measure.

The performance of string similarity metrics has been analyzed in the context of names matching in the available literature. Cohen et al. (2003) analyzed the performance of string similarity metrics in name-matching tasks. They compared the performance of string metrics according to edit distance metrics such as Levenshtein, Jaro-Winkler, Jaro-measure, Needleman-Wunsch, Smith-Waterman and N-gram, token-based distance metrics such as TFIDF, Cosine, Jaccard and hybrid metrics such as Monge-Elkan. They found that the performance measures of Monge-Elkan, TFIDF and Soft TFIDF are the best in each category according to accuracy.

Jimenez et al. (2009) performed some experiments on 12 name matching data sets comparing generalized MongeElkan with three representative character-based string measures: Bigrams, Edit distance and Jaro similarity. They found that the generalized MongeElkan method outperformed the original MongeElkan method when character-based measures were used to compare tokens. In the record linkage problem, the similarity measures are used to quantify the degree of similarity or closeness of two data entities (Koudas et al., 2006). Stoilos et al. (2005) developed a string metric

for ontology alignment and compared this metric to other string metrics such as Levenstein, JaroWinkler, MongeElkan, SmithWaterman, NeedlemanWunsch, 3-gram and sub-string on a subset of the OAEI benchmark test set. They found that the match performance of MongeElkan and SmithWaterman were very poor on their test set.

Using only string similarity metrics do not provide good performance for matching (Cheatham and Hitzler, 2013a). For increasing performance, string text processing techniques are required. The techniques have been divided into syntactic and semantic techniques (Cheatham and Hitzler, 2013a). These are described below:

- **Syntactic technique:** The syntactic technique is based on characters in the strings, and it processes these strings without using outside data sources. Tokenization is an example of this approach.
- **Semantic technique:** The semantic technique requires outside data sources such as a dictionary and a thesaurus. Abbreviation and acronym expansion and synonym are examples of this approach.

The string text processing techniques such as tokenization, synonym lookup, abbreviation expansion, stop word removal, stemming and translation (Cruz et al., 2009; Jean-Mary et al., 2009; Jain et al., 2010a; Li et al., 2009; Lambrix and Tan, 2006; Mitra et al., 1999; Madhavan et al., 2001) are described below:

- **Tokenization:** In tokenization, names of elements are parsed into tokens by a customizable tokenizer using punctuation, uppercase, special symbols and digits. For example, *contactEmail* is split into *contact* and *Email*.
- **Abbreviation and acronym expansion:** Abbreviation expands shortcut into their original form using external knowledge sources. Examples of abbreviation and acronym expansions are $NO \rightarrow Number$ and $UOM \rightarrow UnitOfMeasure$ respectively.
- **Stop word removal:** It is necessary to remove stop words in order to increase performance. Some examples of stop words are *above*, *a*, *am*, *is*, *are*.
- **Stemming:** Stemming is a process that is used to reduce the grammatical differences between words due to verb tense, plurals and other word forms to their

base or root form. An example of stemming is *purchased* → *purchasing*. In this case, the verb form of both words is *purchase*.

- **Translation:** Translation is a process that is used for converting elements to the determined languages using a language tag or a sample of words.

- **Synonym Lookup:** The element pairs can be matched according to their semantics using external resources such as a dictionary, ontologies, user-provided synonym tables, domain and common knowledge thesauri. An example of synonym matching is *telephone* → *contact* where purchase order domain knowledge thesaurus is used as an external dictionary. Common knowledge thesaurus, WordNet (Miller, 1995) is also used for measuring the semantic similarity between elements. WordNet is partitioned into nouns, verbs, adjectives and adverbs, which are organized into synonym sets, each representing one underlying lexical concept. Synonym sets are also called synsets, and these sets are interlinked by different relations such as hypernym, hyponym, antonym, meronym and holonym. For instance, elements *section* and *chapter* are semantically similar and matched using WordNet. Sometimes, element names are composed of multiple words where it is necessary to use other text processing techniques before matching the words using external resources. For example, matching *subjectArea* and *topic* need to tokenize the word *subjectArea* into $\{subject, area\}$, and then *subject* and *topic* are matched using WordNet. Another example, *street* and *hasAddress* needs to remove the stop word *has* before starting matching.

Recently Cheatham and Hitzler (2013a) reported evaluation results of string similarity metrics in ontology alignment problem. Their results showed that 1) the performance of different string similarity metrics varies greatly for some types of ontologies, 2) the impact of string text processing strategies is in many cases unhelpful and in some cases count-productive, and 3) the appropriate string text processing strategies can improve the performance. ASMOV (Jean-Mary et al., 2009) uses tokenization, string equality and Levenshtein distance in the terminological matching. It also uses WordNet and UMLS as background knowledge. Tokenization, string equality and Winkler-based similarity for matching concepts are used in Falcon (Hu et al., 2008).

Research Gaps in the Terminological Matching

- String similarity metrics were also used as a component of many algorithmic schema matchers such as COMA (Do and Rahm, 2002), CUPID (Madhavan et al., 2001), GSMA (Cheng et al., 2005). In the record linkage problem, the similarity measures were used to quantify the degree of similarity or closeness of two data elements (Koudas et al., 2006). However, these researches did not clearly show which string metric and what threshold value worked better for schema mapping before and after using text processing techniques, and how the performance of schema mapping was improved by using different combinations of string text processing techniques.
- The findings of Cheatham and Hitzler (2013a) are very useful because they support the requirements of the context-driven string similarity metrics and text processing strategies. However, their research did not examine how different the threshold setting and different combinations of string text processing strategies impact on the performance of the schema mapping.

In order to fulfill the gaps, I compared names of elements using different string metrics and find the best string metric and threshold value for each schema dataset individually described in Chapter 4. In addition, I used string text processing techniques to process names of elements that contain combined words, abbreviated and synonym words. Finally, I found the best string metric, threshold value and text processing technique for each schema dataset. Further, my research clearly analyzed the match performance differences with various settings: string-metric + tokenization, string-metric + abbreviation, string-metric + synonym, string-metric + abbreviation + tokenization, string-metric + tokenization + synonym and string-metric + tokenization +abbreviation+ synonym.

Due to the various types of heterogeneity of the schema datasets, there is no single best combination (string metric+text processing technique) that can perform well in all matching scenarios. It is necessary to apply some combination approaches that can use several combinations (string metrics+text processing techniques) for improving match performance by overcoming matching errors.

B. Matching by Combination Approaches

Schema/ontology matching can be performed by combination approaches - machine learning and knowledge engineering. These approaches are described below:

Machine Learning Approaches for Matching

Machine learning is a part of Artificial Intelligence (Alpaydin, 2010). It explores the study and construction of algorithms that can learn from past experience and makes predictions on data. It is a computer program that requires efficient algorithms for building mathematical models using the theory of statistics, and for storing and processing the massive amount of data to solve the optimization problem. Once a model is learned, its representation and algorithmic solution for prediction has a requirement to be efficient in terms of space and time complexity.

In machine learning, the term database refers to a collection of instances that are usually fixed-length feature vectors. A learning algorithm takes a dataset and its accompanying information as input and returns a statement representing the results of the learning as output (Frawley et al., 1992). The tasks of machine learning are divided into the following three broad categories (Alpaydin, 2010):

- **Supervised learning:** It generates a function based upon assigned labels, which maps inputs to desired outputs.
- **Unsupervised learning:** No label is provided to the learning algorithms. Unsupervised learning investigates patterns naive to a dataset. There is a structure to the input space where certain patterns occur more often than others. So the goal of this learning is to identify what generally happens and what does not.
- **Reinforcement learning:** A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle), without any explicit instruction about the closeness of outcome to the goal or not.

Machine learning helps to find solutions to many problems in vision, speech recognition and robotics (Alpaydin, 2010). Application of machine learning approaches to

large amounts of data is called data mining (Witten and Frank, 2005). The analogy is that a large volume of raw data is extracted from a mine and processed to reach a small amount of very precious material. Similarly, a large volume of data is processed to construct a simple model with valuable use, which has high predictive accuracy.

Machine learning approaches construct classification models based on training datasets which consist of pairs of elements. They use the models to predict unseen pairs of schema elements as TRUE or FALSE. In this way, the matching task is transformed into a classification task (Ngo et al., 2013). The approaches are promising for predicting elements similarity, and they construct their models with sufficient data.

Machine learning approaches have been used in LSD (Doan et al., 2001) and its extension GLUE (Doan et al., 2002) for instance-based matching. LSD focuses on XML schema matching, and GLUE focuses on matching ontologies based on product catalogs or Web directories. Both systems use machine learning techniques: the Multi-strategy learning approach as a base learner for semi-automatically finding matches between the source schema and the mediated schema, the Naive Bayes technique for classifying text based on labels and attributes as a set of tokens, and the meta learner approach which combines the prediction of base learners and assigns weights to the base learner for finding matching among a set of instances. The systems work in two phases - training and testing. In the training phase, they ask the users to manually map some source schemas onto the mediated schema that is used later for training the base and meta learners. After that, data from every mapped schema are extracted to create training samples for the base and meta learners. In the matching phase, the trained learners are used to map new source schemas onto the mediated schema using the following four steps. First, columns are created for the schema elements for a source database where each column is filled with the corresponding instances. Second, the base learners are applied to the data instances in each column and generate predictions. Then the meta-learner combines the results of base-learners. The prediction converter converts the output of the meta-learner into predictions. Third, the domain handler combines the predictions of the prediction converter with the available domain constraints and outputs the final 1:1 mapping from the data source onto the mediated schema. Fourth, user feedback is integrated to enhance the matching accuracy. The advantage is that the machine learning techniques produce good accuracy. The disadvantages are that 1) training periods are required to make accurate predictions (Do and Rahm, 2002),

and 2) significant manual effort is required to build a mediated schema by providing initial expert correspondences (number of initial expert correspondences are not fixed), manually mapping a few source schemas, optionally constructing valid constraints for the constraint handler, and optionally supplying user feedback. In the systems, the machine learning classifiers are used on the same similarity measures where the meta-learner is a linear regression function, and this function has drawbacks in terms of low quality and extensibility (Duchateau et al., 2008).

For schema-based matching, machine learning approaches have also been used in some systems. Marie and Gal (2008) developed a Schema Matcher Boosting (SMB) on a machine learning approach Boosting - AdaBoost. The Boosting approach consists of iterating weak classifiers over the training set. An advantage of this algorithm is that it can readjust weight given to misclassified schema pairs. However, the main drawback is that some classifiers might give poor results with some scenarios. Thus relying only on one classifier is risky. Duchateau et al. (2008) created a schema matching system MatchPlanner based on a machine learning approach decision tree. The system uses decision tree to select the most appropriate match algorithms. The system inputs a set of schemas and a decision tree that is composed of match algorithms and outputs a list of mappings which are validated by experts to find whether the matching is correct or not. The feedback is used to feed into another decision tree for learning. The advantage of using the decision tree is that the performance of the system is improved since the complexity is bounded by the height of the decision tree. Thus, only a subset of these match algorithms is used during the matching process. The system also provides acceptable results with respect to other matching systems. However, the system needs intensive manual efforts to build a decision tree model based on user input, and the decision trees require initial expert correspondences. Further, the decision trees are not always the best classifiers. They may work appropriately for a given domain, and they may be completely inappropriate for another domain.

Berlin and Motro (2002) developed a system called Automatch for automatically matching schemas using Naive Bayesian. It uses the attribute dictionary for acquiring probabilistic knowledge based on the Bayesian learning. When new schemas are required to be matched, this system uses the knowledge for finding an optimal matching. The advantage of this system is that it can learn continuously when new schemas are found to be matched and combine the information with what has already been

learned. It is then not necessary to re-execute the entire learning algorithm because of the statistical nature of the learning algorithm. However, the problem is in selecting statistical features using a probabilistic learning approach. Dhamankar et al. (2004) proposed a semi-automatic system, iMAP that uses beam search (Russell et al., 1995) to control the search through the space of candidate matches. In this system, machine learning, statistics and heuristics methods are used to evaluate the quality of candidate matches. In the matching process, name matchers and new kinds of domain knowledge (overlap data and mining external data) are used to prune some candidate matches.

Machine learning approaches have been used in some other systems for both schema and instance-based matching. Malform-SVM (Ichise, 2008) is a machine learning based framework for ontology matching that constructs attributes using the word list similarity, concept hierarchy similarity, and structure similarity. This framework processes these attributes using the machine learning approach, Support Vector Machine (SVM), for predicting correct and incorrect mappings. Embley et al. (2004) developed an approach based on learning rules of decision trees for discovering hidden mapping between elements. The approach considers both instance level and schema level matching using object-set and structure matchers. At the instance level, value matching techniques are used, and at the schema level, name and value-characteristics matching techniques are used. In this approach, learning rules of decision trees are used to discover hidden mapping between elements. Here, rules are used to match terms in WordNet. However, the decision trees are not used to choose the best match algorithms. YAM++ (Ngo and Bellahsene, 2012) is an ontology matching system where machine learning techniques, such as decision tree, Support Vector Machine and Naive Bayes are used to combine string similarity metrics to produce mappings at the element-level. Using only one classifier may not produce a good match performance (Rahm and Bernstein, 2001; Do and Rahm, 2002). The YAM++ system uses three matchers and selects the best one for a given scenario to generate good quality results. However, in the system, an appropriate classifier is selected by users, or a default classifier is used to learn over a huge mapping knowledge base.

Knowledge Engineering Approaches for Matching

Knowledge engineering is an approach that requires a high level of human expertise for integrating knowledge into computer systems in order to solve complex problems

(Feigenbaum and McCorduck, 1983). The approach consists of the following activities (phases)(Xu, 2011):

- **Problem Selection:** In this phase, a knowledge engineering approach assesses the difficulty or simplicity of a target problem.
- **Knowledge Acquisition:** The process of acquiring knowledge from human experts accurately and efficiently into a Knowledge Base (KB). This task consists of multiple subtasks. For example, the knowledge acquisition phase is concerned with efficient tools and methodologies that facilitate the capturing of knowledge.
- **Knowledge Representation:** The best way of representing the acquired knowledge into machine-readable form. This phase requires distinguishing between different types of knowledge, such as declarative, procedural or meta, and choosing the suitable representation technique, such as frames, production rules and predicate logic to best facilitate knowledge inference.
- **Implementation:** Translating structured knowledge into computer codes using a conventional or Artificial Intelligence (AI)-oriented programming language. The subtask of this implementation phase is an inference engine, which is responsible for determining a means of using the facts available and a known KB to produce new facts, which lead to a conclusion. This task depends on a particular problem and a programmer's discretion. The inference engine supports two inference strategies: forward chaining (data-driven reasoning) and backward chaining (goal-driven reasoning). The forward chaining approach was designed under an assumption that all facts will be available, allowing the inference algorithm to work forwards through a set of rules in the KB to reach a result (Buchanan and Shortliffe, 1984). The backward chaining approach was designed for domains, where it is assumed that few or no facts are available, and it works by asking appropriate questions to gradually hone in a result (Buchanan and Shortliffe, 1984).
- **Testing:** Verifying and validating inserted knowledge.
- **Maintaining and Evolving:** In this phase, a knowledge engineering approach sustains a KB with up-to-date knowledge.

The knowledge engineering approaches are used to develop Knowledge-Based Systems (KBSs) where human heuristics methods are used. KBSs also known as Expert Systems (ES) which are computer programs designed to perform a task where human experts are unavailable, and also to assist human experts in making their decision more consistent or of an improved quality (Bindoff, 2010). These systems are applied where a more heuristic approach is required for classifications.

The task of knowledge engineering is carried out by highly trained knowledge engineers who are usually computer science professionals. They act as mediators between human experts and the KBS. Among all the phases of knowledge engineering, the knowledge acquisition process is the most difficult part which creates communication difficulties between knowledge engineers and domain experts (Xu, 2011).

Traditional KBS are little more than a collection of facts and rules. The systems can derive new facts with an inference strategy from known facts using rules. They consist of an expert system shell, a KB and an inference engine (Bindoff and Kang, 2010). The tasks are described below:

- **Expert System Shell:** An expert system shell is a computer program through which users can assert and retract facts about the current case, create rules and request an inference for classifications or diagnosis (Buchanan and Duda, 1983). Historically, a text-driven command line is used to enter the information of an expert system shell. However, for effectiveness and user friendliness, a GUI is designed (Bindoff and Kang, 2010).

- **KB:** A KB is the program's storage of knowledge or rules which are used by KBS for inferring some data from the known facts. A KB is represented by a rule-based system (Bindoff and Kang, 2010). The rules are developed by IF [conditions] THEN [conclusion]. For an example, IF (x>15) AND (y<10) THEN ClassA.

- **Inference Engine:** An inference engine is an algorithm which applies available facts in the context of a KB to infer a classification or a diagnosis (Buchanan and Duda, 1983).

Knowledge Engineering approaches have been used for schema-based matching in some systems. SKAT (Semantic knowledge Articulation Tool) (Mitra et al., 1999) is

a hybrid system that combines the results of different matchers. This system uses a rule-based approach for matching schemas semi-automatically. In this system, rules are formulated in a first-order logic, and matching methods are chosen to derive new matches to express match and mismatch relationships. The results of this system are sent to the experts, and they approve, reject and mark the matches or the rules used for matches as irrelevant. Based on the suggestion, this system creates correct rules for matching and stores the knowledge achieved by experts for reuse. A Self-Configuring (Peukert et al., 2012) schema matching system predefines mapping rules: starting, aggregation, rewrite, refine and selection. This system is able to correct a mapping problem to evaluate generated mappings from the schema, ontology and model management domains using rules. However, if the rules produces incorrect mappings, then it is not possible to refine the rules and correct the results. TranScm (Translation System) (Milo and Zohar, 1998) is a rule-based schema matching system. It uses multiple matchers in a fixed order. Each matcher is a rule that is created by name equality, synonyms, homonyms and hypernyms matchers. The match is determined by exactly one matcher per node pair. If no match is found or if a matcher determines multiple match candidates, user intervention is then required to provide a new rule (matcher) or to select a match candidate. This TranScm system requires human intervention for adding multiple rules when a source schema does not match with a target schema, and when a source schema is matched with multiple target schemas.

Limitations of the Combination Approaches

The combination approaches have some limitations. Machine learning approaches generally require a suitable training dataset that should be prepared largely in manual, and the approaches cannot easily change their models without sufficient data. In addition, they require training models to be rebuilt if schema data changes over time. Furthermore, they are domain specific, and their models can not be understood by humans.

In the knowledge engineering approaches, a well-defined training dataset is not required. In these approaches, schema mapping is started for a small number of schema elements by adding rules. These approaches do not require training a model, and they only exploit schema information such as element names, data types, domain constraints and structures, so they are fast (Lee et al., 2007). Their performance is high in certain

types of applications in a specific domain. However, knowledge engineering (that means obtaining knowledge from experts and incorporating it into expert systems) is difficult and time-consuming. The difficulties arise because experts never report on how they reach a decision, rather they justify why the decision is correct. These justifications vary markedly with the context in which they are required. Traditional knowledge engineering approaches cannot use the previous matching effort to solve the matching problem of current ones (Lee et al., 2007). For providing information, maintaining rules and simple knowledge acquisition, Compton and Jansen (1990) proposed Ripple-Down Rules (RDR) based on their experience developing the expert system GARVAN-ES1. There are some distinguishing features of RDR compared to the traditional KBS. The features are described below:

- The knowledge acquisition of the traditional KBS is time-intensive which is called a 'knowledge acquisition bottleneck' problem (Compton et al., 1991). The reason is the difficulty in acquiring knowledge from experts due to their unavailability, unwillingness or inability to articulate all that is relevant to what they know or how they reason. However, RDR resolves the knowledge acquisition problem by shifting the focus to validation of cases that involves the identification of the correct conclusion and salient features, not the knowledge engineering process (Richards, 2009). The knowledge acquisition of RDR is incremental and captured within the context in which it holds. The result is the combination of rule-based and case-based reasoning approaches where the cases provide the context for when the rules are valid, and the rules provide the index by which the cases can be retrieved (Kang and Compton, 1994).

- Most ES approaches require a knowledge engineer to conduct various knowledge acquisition activities, and the approaches analyse structured interviews, protocol analysis, repertory grids, laddered grids and card sorts involving a domain expert from which knowledge would be elicited and encoded by the Knowledge Engineer (Richards, 2009). However, in the RDR approach, knowledge acquisition is performed by a domain expert, not the Knowledge Engineer (Compton et al., 1991). The knowledge acquisition approach of RDR is a result of observing a natural behaviour of experts at the Garvan Institute. It involves looking at a case/problem, making a recommendation, and then if prompted, providing some features of the situation to account for their conclusion (Richards, 2009).

Ripple-Down Rules (RDR)

RDR (Compton et al., 1991) is an approach that allows a user to incrementally build a KB while the system is in use, with no outside assistance or training from a knowledge engineer. RDR terminologies are described below:

a. Attributes and Features

An attribute is a characteristic of a class or a concept. There are four types of attribute: nominal, boolean, ordinal and numeric. A nominal attribute has a finite set of possible values. For example, an attribute *Color* may have three possible values $\{Red, Green, Blue\}$. An ordinal attribute also has a finite set of possible values, and there is order among these values. For example, an attribute *Temperature* may have three values $\{Cold, Mild, Hot\}$. A numeric attribute may have continuous or discrete values. Such an attribute has infinite possible values. However, the value of a real-world attribute is usually in a particular range. For example, the attribute $HeightOfAdult$ has a range of values between 50 cm and 250 cm, and a value of this attribute is not allowed to be outside this range. A *Boolean* attribute has two values $\{TRUE, FALSE\}$. The value of a particular attribute is represented by an attribute-value pair, for example, $Temperature = Cold$. A feature is returned by a function which may have attributes, constants and feature values as parameters. For example, $LessThan(Y, Average(x1, x2, x3))$, $LessThan(X, 5)$ and $(X < 5)$.

b. Cases and Rules

A data case or an input case refers to an object in a domain problem. For example, a data case can be a student record, a text document or a pathology laboratory test result. A data case can be thought of as a frame instance in the frame systems, admittedly with less structural information. A data case consists of a set of features, which are often based on attribute-value pairs. A case can be labelled or unlabelled. An example of a labelled case is COLOR=Green: X=5, Y=3, and the label is COLOR=Green.

c. KB of RDR

The KB of RDR is designed as a binary tree with a rule at each node. A rule consists of conditions and a conclusion, like IF [conditions] THEN [conclusion]. A conclusion is a classification or an interpretation. The value of the conclusion part of a rule can be null. This rule called stop rule, is added to stop the application of the currently fired rule. In this case, if a case satisfies the condition part of this rule as well as its parent, this rule invalidates the parent rule's conclusion. Many rules in RDR are exceptions to other rules. If a rule is satisfied by any case, then the conclusion of the rule will be added to the KB unless no exception rule is satisfied and so on with the exceptions to the exceptions. An exception rule is added if the current case conflicts with the currently fired rule, which has no value for the exception link. The exception rule becomes the exception link of the currently fired rule. An alternative rule is added if the current case does not satisfy the current rule. The alternative rule becomes an alternative link of the current rule. Experts do not need any prior knowledge about the rule structure and how the rule is added to the KB after being satisfied by cases.

In the RDR framework, exception rules are created implicitly. The system asks the expert to justify why a misclassified case is not classified as a particular class (for example, $classA$), but another class (for example, $classB$). The expert justifies her interpretation by selecting relevant features that would classify the misclassified case as $classB$ rather than $classA$. This case is called a cornerstone case. The features selected by the expert forms conditions of the exception rule to the rule of $classA$. Cornerstone cases guide the knowledge acquisition sessions by filtering features, which are available for selection.

d. Inference of RDR

RDR aims to deal with a single classification required for a set of cases. For example, if *bat* is a case, then RDR classifies it as a *mammal*. The RDR inference operation is based on searching the KB represented as a decision list with each decision possibly refined again by another decision list. Once a rule is satisfied by any case, the inference is terminated. The inference algorithm for a single classification RDR is the following:

1. Set lastFiredRule and CurrentRule as null

2. Get exceptionRule of rootRule
3. If exceptionRule is not null, set exceptionRule as currentRule
4. Evaluate inputCase with currentRule
   (a) If inputCase satisfies currentRule, set currentRule as lastFiredRule and get exceptionRule of currentRule
      i. If exceptionRule is not null, set exceptionRule as currentRule and go to 4
   (b) Else get alternativeRule of currentRule
      i. If alternativeRule is not null, set alternativeRule as currentRule and go to 4
5. Stop inference process and return lastFiredRule

e. Knowledge Acquisition of RDR

In the RDR approach, knowledge acquisition is carried out directly by the expert alone with the KBS already in routine use. As real life cases of the domain are fed to the KBS, the expert assesses the following two systems' classifications and act accordingly:

- **Missing Classification:** The expert creates a new rule to add the missing classification.
- **Incorrect Classification:** The expert creates an exception rule to the incorrect classification to correct it.

f. Variations of RDR

There are several versions of RDR method including Single Classification RDR (Compton and Jansen, 1990), Multiple Classifications RDR (MCRDR) (Kang et al., 1995), Recursive RDR (Mulholland et al., 1993) and Nested RDR (Beydoun and Hoffmann, 1997).

Single Classification RDR

RDR is only designed to handle single classification tasks. RDR has been successfully added in many industrial applications such as Labwizard commercial RDR system by Pacific Knowledge Systems, adding intelligence to Web browsers by KMAgent, Help Desk Systems, Sonetto System by Ivis Group, Workflow management system by Yawl Group, EMail Management Assistance (EMMA) system (Richards, 2009). The major success with RDR is PIERS, an expert system used to add clinical interpretations to chemical pathology laboratory reports (Compton et al., 1991).

Multiple Classifications RDR

Multiple Classifications RDR (MCRDR) (Kang et al., 1995) has been developed to solve multiple classifications problems in PIERS system. MCRDR like RDR is based on an assumption that the knowledge an expert provides is essentially a justification for a conclusion in a particular context. To allow for multiple classifications without the use of compound classifications, it is necessary to find a classification then continue searching for other applicable classifications. This consideration necessitates a change in the overall structure and the inference approach used for MCRDR. The KB of MCRDR is designed by an exception based n-tree. Each node of the tree is a rule, and each rule consists of conditions and a conclusion. The format of a rule for both RDR and MCRDR is IF cond1 AND cond2 AND.... AND condN THEN conclusion.

Inference of MCRDR evaluates all nodes at the first level (nodes which have the root as a parent) of the KB. It then evaluates the rules at the next level of refinement for each rule that was satisfied at the top level and so on. The process stops when there are no more children to evaluate or when none of these rules can be satisfied by the case in hand. It thus ends up with multiple paths, with each path representing a particular refinement sequence, and hence multiple conclusions. Ultimately, the deepest satisfied nodes of the KB will be added to the result list (Kang et al., 1995). This process of inference is very similar in many ways to that of a standard depth first search.

MCRDR is used in some domains such as pathology, text/Web document classification, help desk information retrieval and medication review.

Recursive RDR

Recursive RDR (RRDR) (Mulholland et al., 1993) has been developed to solve the configuration style problem of Ion Chromatography (IC). IC is a mechanical task where a device is configured with particular detector settings for analyzing a given substance. Mulholland used RRDR to handle the problem of configuring the device. It has more flexible strategies compared to RDR for handling incomplete and missing information. The method is designed to require eight individual RDR Knowledge Bases (KBs). These KBs are inter-dependent, that means the outcome of a particular KB could change depending on the outcome of other KBs. In order to include the strategy, the RDR structure and inference process of RRDR are different from RDR. However, RRDR has problems with cyclic rules - rules which depend on the existence of a classification, and the same classification may be retracted in spite of being fired. RRDR is also very domain specific.

Nested RDR

Nested RDR (NRDR) (Beydoun and Hoffmann, 1997) is not concerned with more general problems of being able to define a rule based on the presence or absence of a classification. This NRDR approach is only concerned with the idea of intermediate classifications. It does this by allowing the expert to create a KB of the SCRDR for each intermediate concept, which is constructed in a hierarchy. It is also vulnerable to cyclic definitions.

Schema Mapping by RDR

Since the research aim is to find matching relationships (matched or not-matched) of schemas, I chose the Single Classification RDR for the schema mapping research. In the RDR approach, each rule can be either a standard production rule (SPR) or a censored production rule (CPR) (Kim et al., 2012). The success of RDR does not depend on representational differences; rather it largely depends on its distinctive operational semantics on standard production rules (SPR). The RDR method organizes SPR for developing an incremental knowledge-based system. SPR has the form $p- > a$, which is interpreted as "if a case satisfies condition p then do action a". RDR systems,

in general, process cases sequentially. Whenever the current KB suggests incorrect conclusions, new rules are added. Whenever a new rule is created, it is necessary to validate the rule by checking whether or not the future cases are given the correct classifications. If any case is incorrectly classified by a rule, RDR systems then acquire exception rules for this particular rule. As a consequence, the expert directly refines the new rule adding conditions until all incorrect cases are removed. However, it is not easy to construct this kind of rules with resource constraints such as limited time and information. To overcome these constraints, Kim et al. (2012) suggest CPR based RDR. It is used to acquire exceptions when a new rule is created using censor conditions. CPR has the form $p- > a \rightarrow c$, which is interpreted as "if a case satisfies condition p then do action a unless the case does not satisfy the censor conditions c". This approach also can provide multiple cornerstone cases that satisfy the main condition clause (positive cornerstone cases) as well as the censored condition clause (negative cornerstone cases). This approach is useful when it is necessary to verify and validate cases.

I applied CPR based RDR for matching names of the schema elements at the element-level. It can handle schema matching problems of false positive (irrelevant matching among schemas is found to be relevant) and false negative (relevant matching among schemas is found to be irrelevant) (Marie and Gal, 2008) using an incremental knowledge acquisition process. The mechanism of the CPR based RDR approach and experimental results are described in Chapter 4.

The CPR based RDR approach improves performance incrementally for schema mapping. However, in the approach, it is necessary to create rules using schema features for matching schema elements one by one. In contrast, machine learning approaches require rebuilding training models if the schema data changes over time. In order to overcome the limitations of both CPR based RDR and machine learning approaches, it is necessary to apply a hybrid approach. In order to develop a hybrid approach called Hybrid-RDR, I combined a machine learning approach and the CPR based RDR approach. In this hybrid approach, only one classification model is created by a machine learning approach, decision tree, for a small number of schema elements and the Knowledge Base (KB) is then built incrementally by adding rules to solve schema matching problems. In this hybrid approach, the following research issues are addressed:

Reusing Previous Match Operations

Reuse of previous match operations is different and more efficient compared to the reuse of previous match results. Reuse of previous schema mapping information is required for improving effectiveness and efficiency of a schema matching system (Rahm and Bernstein, 2001). Auxiliary information such as a dictionary or a thesaurus and the gold standard values, where users provide the match or mismatch information manually, can be reused. Other information about schemas such as names, data types, keys and constraints can also be reused (Rahm and Bernstein, 2001). Some schema matching systems (Madhavan et al., 2001; Do and Rahm, 2002) reuse synonym tables for building element correspondences. Besides, the elements of new schemas to be matched may be similar to the elements of the schemas where matching has been done if new and matched schemas represent the same application domains. Reusing previous match results significantly reduces match effort when dealing with large schemas consisting of hundreds and thousands of elements. In these cases, large schemas are divided into schemas containing small elements. After this, schema fragments matching is generated, and matching results are stored to reuse for matching other schema fragments (Rahm and Bernstein, 2001).

COMA (Do and Rahm, 2002) and COMA++ (Aumueller et al., 2005) implement the fragment matching approach and reuse the previous matching results. For example, for mapping D1 and D3 schema datasets, the systems search the repository for all applicable mapping paths connecting the datasets D1-D2 and D2-D3 and use the MatchCompose operator for combining matching results produced from D1-D2 and D2-D3. However, reusing the previous matching results has some limitations. First, the elements of new schemas (D1-D3) for which a repository is searched must be the same to elements of matched schemas (D1-D2 and D2-D3), and the elements must be from the same domain. For example, the element *Address* of D1 dataset is matched to the element *customerAddress* of D2 dataset, and the mapping information is stored. To reuse the mapping information, D1 and D3 must have the elements, *Address* and *customerAddress* respectively. Second, the schemas should represent the same domain, e.g., purchase order. Madhavan et al. (2005) introduced corpus-based schema matching and reused the matching results. In their matching, a corpus of schemas is leveraged and a machine learning approach is used to infer that two elements match if they match with the same corpus elements. However, the reuse of matching is for small

schemas as it is limited to an element to element. For large schemas and large corpus, the approach requires a significant amount of manual effort for learning training models and using the models for matching. In order to overcome manual effort as well as a domain specific matching, reusing previous match operations called rules is important to incrementally increase match performance and decrease added rules.

Reusing previous match operations have been implemented in the Hybrid-RDR approach. The match rules are not domain specific. That means, the rules that are created for one domain (e.g., purchase order) can be reused for other domains (e.g., conference, bibliography). Reusing rules helps to reduce the required time to generate rules from the very beginning for processing each dataset. The Hybrid-RDR approach is useful for validating large numbers of schema elements.

Graphical User Interface (GUI) Developments

As fully automatic schema matching is not possible, a user interaction is required for the practicability and effectiveness of a matching system. The Clio (Haas et al., 2005) system developed at IBM supports GUI. However, in this system, considerable manual work is required. The Clio system generates alternative mappings as SQL view definitions based on the value correspondences that are defined by the users. Users can view, add, and remove correspondences between schemas. They can add transformation functions to the correspondences. Finally, the results are transferred to users for checking whether the matching is correct or not. If users do not have proper knowledge, manually defining the correspondences is error-prone and time-consuming. In addition, no linguistic matching technique is used in Clio. Some other systems, such as COMA/COMA++ (Do and Rahm, 2002; Aumueller et al., 2005), YAM/YAM++ (Duchateau et al., 2009; Ngo and Bellahsene, 2012), (Peukert et al., 2011), (Lambrix and Tan, 2006), AgreementMaker (Cruz et al., 2009), and MaF (Martinez-Gil et al., 2012) support GUI. In these systems, it is necessary for users to add false negatives and remove false positives one by one by manually clicking on the relevant schema elements shown in the GUI. Manually solving incorrect classifications require user proper knowledge about the domain, and this manual process is time-consuming, error-prone and expensive.

The Hybrid-RDR approach supports GUI to give the scope to the users to correct

and validate the schema mapping results by an incremental knowledge acquisition process. The detail implementation of this Hybrid-RDR approach and experimental results are described in Chapter 5. In this approach, human experts are required to correct schema matching problems. In order to reduce human involvement, a Simulated Expert-based Hybrid-RDR approach is required. This approach is explained in Chapter 6. In the following section I describe the literature related to simulated experts.

Simulated Experts

It may be difficult to evaluate the KBSs because of the unavailability of experts who have to give their time to the training of the project (Bindoff and Kang, 2010). It may also not be easy to train the same system multiple times when schemas are created over time incessantly, and thus knowledge also changes over time. Furthermore, different experts provide different opinions about the correctness or optimality of the system. In order to overcome this problem, simulated experts in RDR have previously been used for the task of evaluating new expert systems' methodologies (Compton et al., 1995; Kang, 1996).

A simulated expert, e.g., another Knowledge-Based System (KBS), as a source of expertise has been used in order to evaluate knowledge acquisition methodologies, RDR (Compton et al., 1995), MCRDR (Kang et al., 1995) and MCRRR (Bindoff and Kang, 2010) by replacing human experts. Knowledge is acquired from a simulated expert by using these knowledge acquisition methods, and a new KBS is built which should have the same competence as the KBS from which the simulated expert is derived. The simulated expert whose source of expertise is a previously built expert system for the domain is asked about the reason for reaching a particular conclusion. This approach has both an advantage and a disadvantage. The advantage of this approach is that endless repeat experiments can be done, and it is also possible for the experimenter to take complete control over all variables (Compton et al., 1995). The disadvantage of this approach is that a data model is already given or will be very easily derived (Compton et al., 1995).

a. Simulated Expert for RDR

The application of a simulated expert for evaluating RDR methodology has been described by Compton et al. (1995) with respect to three different domains using three datasets: Tic-Tac-Toe, Chess End Game and Garvan thyroid diagnosis from Irvine repository. In the experiment, C4.5 (Quinlan, 1993), INDUCT (Gaines, 1989) and the INDUCT/RDR machine learning approaches (Gaines and Compton, 1992) were used to build the KBSs, and these algorithms were used to construct a simulated expert. The RDR KBS was built by adding rules to correctly classify cases, that were not classified or incorrectly classified. The simulated expert selected conditions from the difference list to go in a new rule, and the conclusion of the new rule was the conclusion specified in the database for that particular case. When no rule was satisfied, all the features of the present case were then included in the difference list. Three levels of expertise: Smartest (S1), Smart (S2) and Dumb (D) were used. In this simulated expert for RDR approach, complex domains are considered, and complete KBS are developed.

b. Simulated Expert for MCRDR

A simulated expert has been used by Kang et al. (1995) for evaluating MCRDR methodology instead of a human expert because it is very difficult to handle various cornerstone cases by a human expert. The role of a simulated expert is to identify important features in the case for justifying the reason for changing the classification of MCRDR. Identifying important features by using a simulated expert is a rule trace of the same case run through another KBS with a high level of expertise in the domain. In the research, INDUCT/RDR machine learning approach has been applied to a total number of cases available for building another KBS. The reason for using INDUCT/RDR is that the size of KBS is much smaller than those produced by other machine learning approaches. In the study, it is not expected that the performance of a simulated expert is better than a human expert. Even with the best simulated expert, randomly selecting conditions from difference lists are required.

In evaluating simulated expert for MCRDR, two different domains - Garvan thyroid diagnosis and Tic-Tac-Toe from Irvine repository were used. The problems of using a single classification for these domains are a large size of KBS and repeated

knowledge acquisition. The MCRDR KBS is built by adding rules to correctly classify cases which are not classified or incorrectly classified. Different types of expertise are applied to choose conditions in the simulated expert for MCRDR. Clever expert selects all conditions from the induct rule trace. Moderate expert selects one condition from the intersection of the difference list and an induct rule trace. Stupid expert selects one condition from the difference list. Clever expert (RDR) selects 4 conditions from the intersection of the difference list and induct rule trace. For handling NULL classification using MCRDR, stopping rules are required. In order to use a simulated expert in this situation, the cornerstone case from the rule giving the incorrect classification is run on induct. Negated conditions are selected from the difference lists which are also in the induct trace. After doing the experiment, the authors found that the clever and moderate experts produced similar size KBs and had the similar performance. The KB produced by these simulated experts was only about three times of the induct KB size.

c. Simulated Expert for MCRRR

Multiple Classification Ripple Round Rules (MCRRR) (Bindoff and Kang, 2010) was developed by preserving all the essential benefits and strategies of the RDR method and by adding the ability to create rules which can use classifications as conditions. It can manage cyclic rule definitions by offering minimal restrictions to when and where the expert may define these rules. This method was successfully applied to a complex configuration task, and it was further evaluated through simulation studies. In this method, it can be easily determined by the node, which must be revisited and re-inferred whenever a classification is added or removed from the result list, by maintaining a list of dependents and dependencies for each classification. In the KB of MCRRR, grouping rules are used to configure or plan tasks. In the experiment, some datasets such as bibtex, emotions, enron, genbase, medical, scene and yeast are used. In the approach, four different types of simulated experts are trained using INDUCT/RDR by employing multi-label machine learning approaches with the binary relevance classifier found in the Mulan (Tsoumakas and Katakis, 2007) extension to WEKA (Hall et al., 2009) and they are applied to the datasets incrementally on a case-by-case basis. Different experts use a different number of conditions, which are identified for each classification. For example, the stupid expert uses 25% of the known

conditions, which makes rules that may cause later errors. Moderate, clever, clever (RDR) experts use 50%, 75%, 100% of the known rule conditions respectively.

d. Applications of Simulated Expert

Lee and Andersen (2009) proposed a simulated expert to be used for day lighting in buildings, particularly for visual comfort, health, productivity and energy consumption. As it is very difficult for a human expert to provide accurate information required to populate a KB for the dynamic nature of daylight and a large number of design variables, the authors use Design of Experiments (DoE) method. The method allows the experimenter to obtain information about the way the independent factors affect a given output. The factors used in the research are window geometry, location and distribution of the facade, shading device geometry, and material properties. In the research, they also use a two-level fractional factorial Resolution V design and perform 128 simulations for design variables on each facade-oriented examination. For combining forward ray tracing with radiosity and shadow volume rendering, they use a simulated engine, the Lightsolve Viewer method (Cutler et al., 2008). The simulated expert can obtain highly specific data and build a comprehensive KB with an expanded set of design variables, which improve designs for a variety of goal complexities.

Cao and Compton (2005) addressed the problem of evaluating knowledge acquisition methods and its difficulties. The problems are: 1) the rule of KBS which is developed by a machine learning approach does not correspond to a human expert; 2) the databases which are used by a machine learning approach are domain specific, but domain variations are not explicitly explained; 3) developing a domain representation is not always required as domain representation is provided by a larger information system, which is linked to many KBS applications. The authors described that the KBS has error components, which cause errors to data that is provided. If the data is covered inappropriately, it is called overgeneralization. The errors occur at the rule level when a simulation is applied for obtaining rules from an expert. In order to handle the errors, the authors proposed a parameterized framework based on data model theory to quantify the levels of expertise. In the framework, the domain is assumed to be made up of disjunctive regions where the minimum number of rules required is the number of disjuncts in the domain. The framework is applied to simulate three variants of RDR: Single Classification RDR, Flat RDR and Composite RDR. The result of

simulation shows interesting findings of the relations between a level of expertise and the performance of resulting KBs.

Simulated Expert for Schema Mapping

In order to maintain a KB of large schema datasets reducing human involvement, it is necessary to develop a Simulated Expert-based Hybrid-RDR approach for schema mapping. In this approach, C4.5 (Quinlan, 1993) machine learning approach is used to build the KBSs, which were used previously for the simulated expert (Compton et al., 1995). The RDR KBS is built by adding rules to correctly classify those cases, which are incorrectly classified. In order to achieve this, the simulated expert selects conditions from the difference list to go in a new rule, and the conclusion of the new rule is the conclusion specified in the database for that particular case. When no rule is satisfied, all the features of the present case are included in the difference list. Though the simulated expert is not as creative or wise as a human expert (Bindoff and Kang, 2010), it can be used to assess a knowledge acquisition methodology of RDR at various levels of expertise. In this hybrid approach, a complete KBS is developed. This approach significantly reduces human effort for creating rules when a machine learning approach generates incorrect classifications. Finally, the matching results are sent to the users for checking, and the rules are added to the KB to be reused whenever the schema data changes over time. This approach is reuse-oriented with less human intervention. This approach makes the schema matching system as automatic as possible. The detailed algorithm and description of the application of the Simulated Expert-based Hybrid-RDR for schema mapping is described in Chapter 6.

ii. Constraint-based Approaches

Constraint-based matching approaches are used to overcome internal constraints, which are applied to the definitions of elements such as datatypes, keys and cardinality (Shvaiko and Euzenat, 2005). Data types include string, integer, varchar and date. Keys can be primary, unique and foreign. Constraint-based matching is required in measuring similarities between schemas if the schemas contain the constraint information (Rahm and Bernstein, 2001). For example, the data type and key of an element *empName* of one dataset are varchar and primary key respectively, and the data type

and key of an element *ename* of another dataset are string and unique respectively, then the elements are matched because the string is equivalent to varchar, and the primary key is equivalent to unique. Matching elements based on constraints require a compatibility table for automatically looking up the similarity between different occurrences of a constraint. Some matching systems use the technique. Among them, SMB (Marie and Gal, 2008) considers name and domain constraints for matching schemas. CUPID (Madhavan et al., 2001) combines names, data types and constraints at the finest levels of granularity. I do not consider constraint-based matching to reduce the complexity of schema matching.

3.3.2 Structure-level Approaches

The results of element-level matching are used for structure-level matching to identify the structural similarity between a pair of nodes from two schemas/ontologies by analyzing positions of nodes on the hierarchical structure of graphs. Structure matching is used to adjust incorrect matches found from element-level matching. The matching considers different types of relationships, such as containment relationships, and is-a/part-of relationships depending on the modeling capabilities of the schema language. In order to identify different kinds of structurally related elements for matching, schema elements and their relationships are represented in a directed or undirected graph. The matching can be performed by determining neighbourhood similarity and iterative matching (Hai, 2005). Different kinds of neighbour elements, such as parents, children and leaves can be considered to estimate the similarity between two elements structurally. Madhavan et al. (2001) described the following intuitions for structure-level matching:

1. Atomic elements (leaves) in the two trees are similar if they are individually (linguistic and data type) similar, and if elements in their respective vicinities (ancestors and siblings) are similar.

2. Two non-leaf elements are similar if they are linguistically similar, and the subtrees rooted at the two elements are similar.

3. Two non-leaf schema elements are structurally similar if their leaf sets are highly similar, even if their immediate children are not. This is because the leaves

represent the atomic data that the schema ultimately describes.

The above intuitions are used in COMA (Do and Rahm, 2002) and Cupid (Madhavan et al., 2001). Melnik et al. (2002) presented a similarity propagation algorithm called Similarity Flooding (SF) and explored its usability for schema matching. This algorithm works based on the following intuition. First, schemas are converted into directed labelled graphs. These graphs are used in an iterative fixed point computation to determine the matches between corresponding nodes of graphs. This algorithm uses the concept that two nodes are similar when their neighbor elements are similar. This algorithm is used in some systems (Kirsten et al., 2011; Ngo and Bellahsene, 2012; Wang, 2011) for matching the hierarchical structure of a full graph.

Other algorithms such as role similarity analysis (Martinez-Gil et al., 2012), and structural proximities, clustering and GMO (Hu et al., 2008) are used for doing structure-level matching. The similarity between nodes can be computed based on their relations (Maedche and Staab, 2002). In S-Match (Giunchiglia and Shvaiko, 2003), semantic structure matching is implemented by a decider propositional satisfiability (SAT). The algorithm works only on Directed Acyclic Graphs (DAG's) and is-a links. SAT deciders are correct and complete decision procedures for propositional logics. SAT allows only and all possible mappings between elements to be found. In SKAT (Mitra et al., 1999), two nodes are matched at the structure-level if their parent nodes and children nodes are matched. Anchor-Flood (Seddiqui and Aono, 2009) uses internal and external similarities and iterative anchor-based similarity propagation for structure-level matching. In ASMOV (Jean-Mary et al., 2009), a weighted sum of a domain and range similarities are used for computing an iterative fixed point, and measuring hierarchical and restriction similarities.

Another structure-level matching is to restrict a matching scope in an iterative matching can be performed by traversing schemas either top-down or bottom-up (Hai, 2005). Especially, the similar elements identified at one level determine the scope for the next level, which then considers either their descendants in the top-down approach or ascendants in the bottom-up approach. A top-down approach matches at a high level of a schema structure restrict choices for matching finer grained structure only to those combinations with matching ancestors. For this, this approach is less expensive than the bottom-up approach. However, sometimes though finer-grained elements match well, the top-down approach can be misled if top-level schema structures are

very different. This top-down approach is used in COMA++ (Madhavan et al., 2005). Inversely, a bottom-up approach considers matching schema structure by comparing all combinations of fine-grained elements.

In general, structure matching of schemas/ontologies can be conducted depending on consideration of different levels of information represented by schemas/ontologies elements. Considering only one context (parent or leaf or child) does not provide appropriate results. I use the well-known Similarity Flooding algorithm considering three neighbouring contexts: parent, children and leaf for structure-level matching. I show the experimental results in Chapter 5.

3.3.3 Aggregation Functions

The mappings discovered from element-level matching and structure-level matching are combined by using some aggregation functions. I define the similarity values found from element-level matching and structure-level matching by esim and ssim respectively. The aggregation functions are described below:

- **Harmonic Mean:** Harmonic mean is calculated by the following function:

  Harmonic mean=2*esim*ssim/ (esim+ssim).

  This combination strategy is used in some schema/ontology matching systems (Do and Rahm, 2002; Ngo et al., 2011a).

- **Average:** The average similarity is calculated by dividing the sum of the similarity values of two string metrics for each name pair by the total number of similarity functions. Average value is calculated by the following function:

  Avg= (esim+ssim)/2.

  This combination strategy is used in some schema and ontology matching systems (Do and Rahm, 2002; Volz et al., 2009; Jimenez et al., 2009).

- **Minimum:** This strategy returns the minimum similarity value between two string metrics. Minimum value is calculated by using the following function:

  Min=Math.min (esim, ssim).

  This combination strategy is used in some schema and ontology matching systems (Do and Rahm, 2002; Volz et al., 2009; Massmann and Rahm, 2008).

- **Maximum:** This strategy returns the maximum similarity value between two string metrics. Maximum value is calculated by using the following function:

  Max=Math.max (esim, ssim).

  This combination strategy is used in some matching systems (Do and Rahm, 2002; Volz et al., 2009; Massmann and Rahm, 2008).

- **Weighted:** This strategy returns a weighted sum of similarity values. The value found from structure-level matching is used as a threshold value, which is a weight of element-level matching, and a weight of structure-level matching, W_struct is (1-threshold) (Ngo et al., 2011a). The weighted similarity of the element pair e1 and e2 is calculated as

  wsim(e1,e2)=W_struct.ssim(e1,e2)+(1-W_struct).esim(e1,e2)

  This combination strategy is used in some matching systems (Do and Rahm, 2002; Ngo and Bellahsene, 2012; Madhavan et al., 2001).

3.4 State of the Art Schema and Ontology Matching Systems

In the following, some state of the art schema and ontology matching systems are described based on the classifications provided in Section 3.1.

YAM. YAM (Yet Another Matcher) (Duchateau et al., 2009) is a machine learning based dedicated schema matching system used for data integration. This system takes XML schemas, Web interfaces, semi-structured or structured data models as input and produces 1:1 mapping between schemas as output. It works in two phases: a learning phase and a matching phase. At the learning phase, YAM considers user requirement such as a preference for recall or precision. For this, expert correspondences, weights and threshold are provided in this phase. This system uses a KB that consists of 20 classifiers from the Weka (Hall et al., 2009) library, including decision trees (J48, NBTree, etc.), aggregation functions (SimpleLogistic), lazy classifiers (IBk, K*, etc.), rule-based approaches (NNge, JRip, etc.) and Bayes Networks, 30 string similarity metrics including all from the secondstring[1] project, and pairs of schemas which have

[1] http://secondstring.sourceforge.net

already been matched. For matching, the KB built at the learning phase is used to match unknown schemas that produce a list of correspondences between schemas. Finally, the discovered correspondences are validated by the users. However, in this system, an appropriate classifier is selected by users or a default classifier learned over a huge mapping knowledge base is used.

This system runs many iterations until the similarity scores between elements become stable, and it removes some incorrect mappings (pre-defined). However, it takes considerable time for many iterations, and it requires a training model to be rebuilt if schema data changes over time. In this system, users are asked to select appropriate classifiers. If users do not have proper knowledge, then they depend on default classifiers. However, default classifiers often do not produce a good match performance. In addition, the YAM system considers user requirements such as a preference for recall or precision, and this system requires expert correspondences. Without proper knowledge, it is not easy to provide the preference between precision and recall. As the system requires initial expert correspondences, so KB is not initially empty.

SAMBO. SAMBO (System for Aligning and Merging Biomedical Ontologies) (Lambrix and Tan, 2006) is an instance-based matching system that matches and merges biomedical ontologies. It handles ontologies in OWL format and generates 1:1 alignments between concepts and relations. It uses similarity-based matchers including terminological, structural (concept to concept) and background knowledge based (UMLS and WordNet). For terminological matching, it uses n-gram and edit-distance to match a list of words. The results are produced by the combination of all the matchers based on users' defined threshold. For structure-level matching, it considers two concepts are similar with respect to is-a or part-of hierarchies relative to already matched concepts. This system works in two phases: training and matching. In the training phase, concept classifiers and Naive Bayes are learned for the available instances of ontology. In the matching phase, the learned concept classifiers are applied to the instances of another ontology in order to determine the concepts an instance is predicted to belong to. For deriving concept similarities and concept correspondences, the instance-concept associations aggregated by the Jaccard-based set similarity measure, are used. Finally, results are sent to users for rejection, approval and modification manually. After completing matching, this system can merge the matched ontologies, compute the consequences and check the newly created ontology for consistency. How-

ever, SAMBO can classify small ontology sizes between 10 and 112 concepts (Rahm, 2011). This system is used for ontology merging.

Self-Configuring Peukert et al. (2012) proposed a rule-based Self-Configuring schema matching system. This system inputs XML schema trees, ontologies, database schemas or meta-models and outputs 1:1 mapping between schemas. It uses different terminological matching criteria such as name, data types, annotations, and synonyms using WordNet. For structure-level matching, it uses PathMatcher rule. This rule relies on the PathVariance, SchemaDepth, Selectivity and MultiMatches features. This system is able to correct mapping problems to evaluate generated mappings from a schema, an ontology and a model management domain. It depends on features that are computed from input schemas and intermediate mapping results. The features are then used in matching rules to select matchers, aggregation and selection operators. The rules represent expert knowledge on how to define or adapt schema matching processes. The matching process is iteratively extended, rewritten and executed to correct matching problems. This system is used for data integration, ontology alignment or model transformation.

However, in this system, mapping rules such as starting, aggregation, rewrite, refine and selection are predefined. The system therefore faces problems when the viewpoint of two schemas is highly different. The second problem is that if some pre-defined mappings are incorrect, and these methods are run only one time to produce new mappings, then the accuracy of new results will be unconfident. The third problem is that the system tunes matching processes manually, and it does not split the process control flow based on the type of elements to be matched.

MaF. Martinez-Gil et al. (2012) developed an ontology matching framework, MaF, that takes two OWL ontologies as inputs and produces 1:1 mapping between ontologies. It performs both element-level matching and structure-level matching at both instance-level and schema-level. For element-level matching, the framework uses concept similarity analysis (CSA2) algorithms including distance-based, name-based and WordNet-based methods. It uses Role similarity analysis (RSA2) algorithms including class, object property and data type property methods for structure-level matching. It also uses hybrid Similarity analysis (HSA2) algorithms that combine CSA2 and RSA2 for mapping between ontologies. In addition, it combines different mapping

results by using average, maximum, minimum, Murkowski distance, weighted product and weighted sum aggregation functions. In order to obtain the most promising mapping result, this framework allows users to filter mapping results using hard, delta and proportional thresholds. It uses the largest number of algorithms, and it can test the largest combinations of algorithms. However, the framework is user-dependent, and users need to gather experience to select the appropriate algorithm for mapping. The framework does not allow users to correct and validate the result if the selected algorithm produces incorrect matching results.

OMReasoner. OMReasoner (Shen et al., 2014) is an another ontology matching system that combines multiple individual matchers: string similarity metrics (prefix, suffix, edit distance) for syntactic matching, an external dictionary (WordNet) and description logic (DL) reasoner for semantic matching, and constraint-based matching techniques. The multiple matching results are combined by the weighted summarizing algorithm (WeightSum) and maximum method (Max). The OMReasoner system has some limitations. 1) It uses a threshold value in the syntactic matching to determine whether the similarity is regarded as equivalence. However, applying a specific threshold value for all ontologies is not feasible. 2) It does not use preprocessing techniques such as stemming and tokenization, which are necessary to eliminate a specific character and separating compound words respectively. 3) It does not use comments and labels of concepts for matching. 4) It employs a description logic (DL) reasoner with external rules to reason about ontology matching. However, the reasoning technique is time-intensive, and it does not have a large impact on results. Finally, it does not consider structure-level matching, which is required to match the hierarchical context of concepts.

XMap++. XMap++ (eXtensible Mapping) (Djeddi and Khadir, 2014) system matches two ontologies using string, linguistic-based and structural-based similarity matchers. This system uses the cosine similarity as a string similarity measure to match labels, names and identity of two ontologies. Here, the Bing Translator is used for translating labels with different languages. WordNet is used as a linguistic matcher for matching words semantically. The system also loads WordNet fully into memory to reduce time for matching large ontologies. For structure-level matching, it uses adjacency relationships (subClassOf and is-a) to correct the elements, which are not matched by string and linguistic matchers. Aggregation operators: aggregation,

selection and combination are used for combining similarity values produced by three matching techniques. Finally, a threshold value is applied to filter the results. However, this XMap++ system does not take into account comments of schema elements and instance information. In addition, the system cannot avoid multiple accesses to the Microsoft Translator within the matching process.

AOTL. AOTL (Ontology Alignment at Terminological and Linguistic level) (Khiat and Benaissa, 2014) is an ontology matching system that performs matching only at the element-level. It takes two ontologies and extracts their elements names, labels, properties and instances. It calculates similarities between elements using string similarity algorithms: Levenshtein-distance, block-distance, Jaro, SLIM-Winkler, Jaro-Winkler, Smith-Waterman and Needleman-Wunsch at the terminological level and the external resource WordNet as the linguistic resource. It represents the calculated similarities in a matrix and applies a filter to identify alignments. However, choosing a specific threshold value for all ontologies is not feasible as different ontologies may have different structures, so a specific threshold may not work for all ontologies. The system only considers element-level matching. It does not consider structure-level matching, which is necessary to match the hierarchical context of elements.

MassMtch. MassMtch (Schadd and Roos, 2014) performs both element-level and structure-level ontology matching. In order to match elements, it uses 3-grams, Jaccard and WordNet at the element-level, and hybrid similarity at the structure-level. Virtual document similarity that uses a weighted combination of descriptions of concepts is applied in the system. All the similarities are combined using an average aggregation function. Finally, the Naive descending extraction algorithm is applied to the aggregated similarity matrix for determining the final mappings. However, the system has run time and memory problems for large ontologies.

3.4.1 Summary of Schema and Ontology Matching Systems

The summary of the approaches is given in Table 3.2. The *System* row presents names of schema/ontology matching systems. The *Input* row represents the input format used by the systems. The *Level* row describes whether the systems do mapping at the schema or instance or both levels. The *Match cardinality* row provides information about the cardinality of the computed alignment produced as output by the systems.

The systems which have GUI and those which do not are represented in the *GUI* row. The combination approaches used in the systems are represented in the *Approach* row. The rows *Terminological matching* and *Structural matching* describe the matching approaches used on element names of schemas/ontologies and hierarchical graphs respectively.

Summary of Analysis

Analytical summary of the schema and ontology matching systems explained in Table 3.2 are given below:

- Most of the systems provide both element-level matching and structure-level matching, but YAM, OMReasoner and AOTL provide only element-level matching.
- Some systems support multiple schema types, and some systems support only one schema type as input.
- Most of the systems focus on producing 1:1 matching cardinality as output.
- Most of the systems consider only schema-based matching. Some systems, YAM and MaF, consider both schema-based and instance-based matching.
- Some systems use rule based and machine learning based approaches. Some systems do not use any combination approach.
- Of all the systems, YAM, SAMBO and MaF are equipped with the GUI.

More systems are described in appendix.

I have already described the limitations of the existing schema and ontology matching systems. In order to overcome the limitations of the above systems, it is necessary to build an incremental KSMS that performs matching both at the element-level and structure-level. The system is described in Chapter 5.

3.5 Summary of this Chapter

In this chapter, I described schema and ontology matching systems according to the classifications: instance-based, schema-based, instance and schema-based, usage-based, element-level and structure-level. I also provided comparisons among some systems based on features, showing that combination approaches such as machine learning and knowledge engineering are used in the schema/ontology matching systems. The combination approaches have both advantages and limitations. In order to exploit the advantages and overcome the limitations, I proposed a hybrid approach that combines both machine learning and knowledge engineering approaches. In addition, I summarized the state of the schema/ontology matching approaches based on some features: type of input, schema or instance-level, output or match cardinality, GUI, terminological matching approaches, combination approaches and structure matching strategies. I also found some limitations in the schema/ontology matching systems. In order to overcome the limitations, I proposed a KSMS for schema mapping.

Table 3.2: Summary of schema and ontology matching systems

System	**YAM (Duchateau et al., 2009)**	**SAMBO (Lambrix and Tan, 2006)**	**Self-Configuring (Peukert et al., 2012)**	**MaF (Martinez-Gil et al., 2012)**
Input	XML schemas, semi-structured or structured data models	OWL	XML, ontologies, database schemas	OWL
Level	schema, instance	instance	schema	schema, instance
Match cardinality	1:1	1:1	1:1	1:1
GUI	yes	yes	no	yes
Approach	20 classifiers from Weka	Naive Bayes	rule	-
Terminological matching	30 string similarity metrics	String similarity metrics, WordNet	name, WordNet, datatypes, annotation, instance	concept similarity analysis (CSA2) algorithms
Structure matching	-	similarity based on is-a, part-of hierarchies	PathMatcher rule	role similarity analysis (RSA2) algorithms

System	**OMReasoner (Shen et al., 2014)**	**XMap++ (Djeddi and Khadir, 2014)**	**AOTL (Khiat and Benaissa, 2014)**	**MassMtch (Schadd and Roos, 2014)**
Input	OWL	OWL	OWL	OWL
Level	schema	schema	schema	schema
Match cardinality	1:1	1:1	1:1	1:1
GUI	no	no	no	no
Approach	-	-	-	-
Terminological matching	prefix, suffix, edit distance, WordNet	cosine, WordNet, Bing Translator	string similarity metrics	3-grams, Jaccard, WordNet
Structure matching	-	adjacency relationships (subClassOf and is-a)	-	hybrid similarity

Chapter 4

Basic and Combination Approaches and Experiments

In Section 4.1, I explain that schema matching at the element-level is performed by terminological matching approaches and combination approaches. I represent the aims of the evaluation of the terminological schema matching approaches and combination approaches in Section 4.2 and Section 4.3 respectively. In Section 4.4, I describe these approaches. I describe schema datasets and experimental procedures in Section 4.5. I thoroughly examine how string similarity metrics and text processing techniques impact on the performance of the terminological schema matching and highlight their limitations. I also describe evaluation approach, experimental design and evaluation results in this section. Finally, I conclude in the last section.

4.1 Introduction

Schema mapping is a high-level specification of the relationships between pairs of schema elements of two schemas called a source schema and a target schema (Cate et al., 2013). Schema matching is a process that identifies semantic mappings/correspondences between related elements of schemas (Bellahsene et al., 2011). Most mapping research has been carried out among database schemas, XML-schemas, and ontologies. In this chapter, I focus on schema mapping in XML-schemas on the Web. Schema mapping can be conducted by schema matching systems that combine different matching algorithms

with a mapping selection module (Ngo et al., 2013). In Chapter 3, I described schema matching is performed at two levels: element-level and structure-level. While element-level matching considers only matching names of elements, structure-level matching uses the results of element-level matching for matching the hierarchical structure of a full graph. Element-level matching is performed by the terminological matching approaches and combination approaches. Terminological matching is the basic technique that compares element names using string similarity metrics and text processing techniques to decide which elements are matched. The results of element-level matching mostly impact on the performance of structure-level matching (Al-Ghanim et al., 2011). In this chapter, I explain the evaluation process of element-level matching.

I define the schema mapping process as: $\mathcal{M} = c(S_i, T_j, V_{i,j})$, where S_i is a source schema, T_j is a target schema, $V_{i,j}$ is an attribute value vector ($V_{i,j} = \{v_1, v_2, , v_n\}$) and $\mathcal{M}$ is a boolean output (if S_i is matched with T_j, return true; otherwise return false). The attribute values are derived by applying different similarity functions to S_i and T_j, and to the values of S_i and T_j after text processing (e.g., synonym, tokenization, and abbreviation expansion). The schema mapping problem is therefore to find the classification function (c) that accurately predicts a real relation between two element pairs.

String similarity metrics calculate the degree of similarity between two strings. It is well known that the match performance of string metrics varies among different data. More detailed explanations of metrics and their evaluation results are described in some research (Cohen et al., 2003; Cheatham and Hitzler, 2013a). String similarity metrics produce a numeric value ranging from 0 to 1 in normalised similarity metrics. It is therefore necessary to set a threshold value for deciding whether or not a source schema element and a target schema element are matched. Various text processing techniques such as tokenization, abbreviation and acronym expansion, and synonym processing have been suggested to improve schema mapping performance (Cheatham and Hitzler, 2013a). So performance checking of the terminological matching approaches is required.

4.2 Aims of Terminological Schema Matching

The aim of my research is to analyze the terminological matching performance with different parameter settings. In particular, I examine the following two problems. *First*,

I examine the threshold setting strategy. The performance of different string metrics (s) is compared with different similarity threshold (t) settings. Generally, if the higher threshold is set, higher precision and lower recall can be obtained. In line with this, many researchers have tried to find an optimal threshold and apply this to any ontology mapping problem (Cheatham and Hitzler, 2013a). However, I assume that the optimal threshold values are different according to the schema mapping context. *Second, I examine the text processing strategy.* The performance of the terminological matching was evaluated with different text processing techniques (p). The performance of individual text processing technique was previously examined by Cheatham and Hitzler (2013a), but they did not examine the impact of the combination strategies of the text processing techniques. Furthermore, they did not examine how text processing strategies can be changed according to the mapping context. Examination of these problems is important because if threshold setting and text processing strategies are dependent on the context, it implies that a context-dependent schema matching system is essential for schema mapping.

Different string similarity metrics and text processing techniques perform well for different schema elements. This is because schema elements contain different characteristics such as the frequency of identical, abbreviated, synonym and combined words. It is therefore necessary to combine these techniques effectively by combination approaches - machine learning and CPR based RDR.

4.3 Aims of Combination Approaches

The aim of this research is to improve match performance by solving schema matching problems of false positives and false negatives. I perform element-level matching by the following ways. *First, I use CPR based RDR for schema mapping.* CPR based RDR can handle schema matching problems using an incremental knowledge acquisition process. *Second, I use machine learning approaches for schema mapping.* Machine learning approaches usually reconstruct their models with sufficient data for handling matching problems. *Finally, I compare the performance of machine learning approaches to CPR based RDR.*

4.4 Methodology

In this section, I describe the terminological matching and combination approaches.

4.4.1 Terminological Matching Approaches

Terminological matching is performed by string similarity metrics and text processing techniques.

i. String Similarity Metrics

String similarity metrics are described in Chapter 3.

ii. String Text Processing Techniques

String text processing techniques are described in Chapter 3.

4.4.2 Combination Approaches

The combination approaches - CPR based RDR and machine learning are described below:

i. CPR based RDR Approach

In order to validate a large number of cases, rules of RDR have been modified as ordered CPR (Kim et al., 2012) where censored conditions are added to create censor rules. In the CPR based RDR approach, a rule is fired as long as none of its censor rules are fired. This approach is shown in Figure 4.1.

This approach is described below according to Kim et al. (2012):

KB of CPR based RDR: The KB of CPR based RDR is designed as an n-ary tree. An example of KB is described in Figure 4.2.

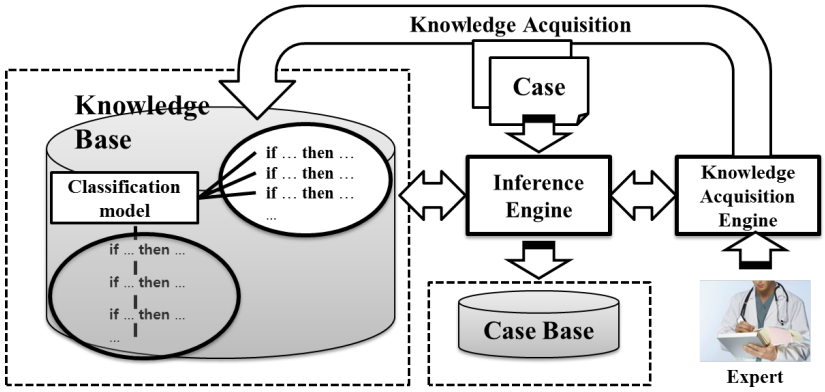

Figure 4.1: CPR based RDR approach

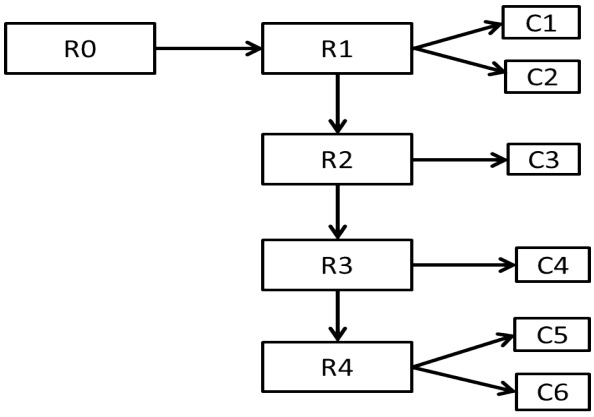

Figure 4.2: An example of KB of the CPR based RDR approach

Each node of the tree is a rule and each rule consists of IF [condition] THEN [conclusion] UNLESS [censor-condition]. A conclusion is a classification or an interpretation. The value of the conclusion part of a rule can be NULL. This rule called stop/censor/exception rule is added to stop the classification of the currently fired rule. Many rules in CPR based RDR are added as censor rules of other rules to classify the incorrectly classified cases as NULL classifications. Other rules called alternative rules are added to make these NULL classifications as correct classifications.

In Figure 4.2, the KB is initially empty. R0 (rule 0) is defined as a root rule. It is an entry point of the inference process, and it is always true. It performs default checking for checking the validity of the case processed. The first level rules are denoted by R1, R2, R3 and R4, where R2, R3 and R4 are called the alternative rules of R1. The censored rules are denoted by C1, C2, C3, C4, C5 and C6. These censor rules are added as censor nodes/children rules to the KB, and the alternative rules are added as parent rules to the KB. The rules are created using attributes, operators (<,>,==,!=,<=,>=) and values.

Inference of CPR based RDR: The inference process is based on searching the KB represented as a decision list with each decision possibly refined again by another decision list (Kang et al., 1995). Once a rule is satisfied by any case, this process evaluates whether or not the censor conditions are matched to the given case. If any censor rule is not satisfied, the process then stops with one path and one conclusion. However, if any censor rule is satisfied, other rules below the rule that was satisfied at the top level is evaluated. This process stops when none of the rules can be satisfied by the case in hand.

Knowledge Acquisition: Knowledge acquisition is a process that transfers knowledge from human experts to Knowledge-Based Systems (KBSs) (Kang et al., 1995). Knowledge acquisition of CPR based RDR is required to handle the incorrect or missing classifications. The knowledge acquisition process has been divided into three parts. Firstly, the system acquires the correct classifications from the expert. Secondly, the system decides on the new rules' locations. Thirdly, the system acquires new rules from the expert and adds them to correct the KB. If the current KB suggests incorrect classification, it is necessary to add a censor rule for making the classification NULL.

If the current KB suggests no classification for any case, a new rule is added as an alternative rule. This rule is added as a child rule of the root node of the KB. The incremental knowledge acquisition algorithm (Compton and Cao, 2006) is described in Algorithm 1:

Algorithm 1: Algorithm for Incremental Knowledge Acquisition

1. Start with an empty KB;
2. Accept a new data case;
3. Evaluate the case against the KB ;
4. If the result is not correct, an expert is consulted to refine the KB;
5. If the overall performance of the KB is satisfactory, then terminate, otherwise go to Step 2;

Cornerstone Case (CB): The cases used to create rules are called cornerstone cases, and these cases are used in consequent knowledge acquisition (Compton and Jansen, 1990). The only difference between conventional RDR and CPR based RDR in managing cornerstone cases is that the CPR based RDR approach maintains non-conforming cases as well as conforming cases for creating a new rule while the conventional RDR approach only maintains conforming cases for a new rule.

ii. Machine Learning Approaches

In this research, I use 11 machine learning approaches: decision trees such as J48, Random Forest, REPTree, BFTree, ADTree, Functional Tree, SimpleCart, and NBTree, rules such as Decision Table, meta such as AdaBootM1, and Bayesian Network, Naive Bayes. These are supervised classification approaches. These approaches input a collection of records (training set) where each record contains a set of attributes, and one of the attributes is the class. The purpose of training a dataset is to build up a model for the class attribute as a function of the values of other attributes. A test set is used to determine the accuracy of the model.

- **J48.** J48 classifier is a simple C4.5 (Quinlan, 1993) decision tree for classification. It divides a dataset into smaller subsets and creates a binary decision tree with decision nodes and leaf nodes. A decision node has more than one branch, and

the top most decision node corresponds to the best predictor called the root node. A leaf node represents a classification or a decision. For a given dataset, one or more decision rules are created that describe the relationships between inputs and targets. These rules can predict the value of new or unseen cases if the cases match with the inputs.

- **Random Forest.** Random Forest (Breiman, 2001) consists of many individual trees to operate quickly over large datasets. In order to build each tree in the forest, the forest can be varied by using random samples. A tree is constructed as (Pater, 2005): (1) bagging takes samples from datasets with replacement for a training set and selects a small amount of data to be used for tree construction, (2) a random number of attributes is chosen from the bagging data and the one with the most information gain is selected to comprise each node, (3) all the nodes of the tree are traversed until no more nodes can be created due to information loss, and (4) bagging error is estimated by running a dataset through tree and measuring its correctness.

- **ADTree.** ADTree (Alternating decision tree) (Freund and Mason, 1999) is a generalization of decision trees, voted decision trees and voted decision stumps. It consists of decision nodes and prediction nodes, where a decision node specifies a predicate condition and a prediction node contains a real-valued number. It always has prediction nodes as both root and leaves. In ADTree, an instance defines a set of paths. The instance is classified by following all paths for which all decision nodes are true and by summing any prediction nodes that are traversed.

- **REPTree.** REPTree (Reduced-Error Pruning tree) is a fast tree learner that uses information gain/variance reduction for building a decision or regression tree and prunes it using reduced-error pruning (Witten and Frank, 2005). Like C4.5, it can split instances into pieces in order to handle missing values. In this approach, it is possible to set some parameters such as the minimum number of instances per leaf, maximum tree depth, minimum proportion of training set variance for a split (numeric classes only), and a number of folds for pruning.

- **SimpleCart.** SimpleCart is a decision tree learner for classification that uses minimal cost complexity pruning strategy of CART (classification and regression tree) (Loh, 2011). For classification, it is possible to set some parameters such

as the minimum number of instances per leaf, the percentage of training data used to construct the tree, and the number of cross-validation folds used in the pruning procedure (Witten and Frank, 2005).

- **NBTree.** NBTree (Kohavi, 1996) is a hybrid classification approach that combines decision trees and Naive Bayes. It creates trees with leaf nodes that are Naive Bayes classifiers. It selects a threshold-like decision tree using a standard entropy minimization technique. It uses cross-validation for estimating accuracy using Naive Bayes.

- **FT.** FT (Functional Tree) (Gama, 2004) is a supervised approach for classification and regression problems. For prediction problems, it uses functional nodes as a bias reduction process and functional leaves as a variance reduction method. It uses linear functions both at the decision nodes and leaves, that facilitate large datasets.

- **BFTree.** BFTree (Bloom Filter Tree) (Athanassoulis and Ailamaki, 2014) is produced by employing probabilistic data structures to trade accuracy for size. It consists of a root node and an internal node. For offering competitive search performance, it exploits pre-existing data ordering or partitioning.

- **Decision Table.** Decision table represents a conditional probability table, and it stores input data based on a selected set of attributes (Hall and Frank, 2008) where each element is associated with a class probability. In learning decision table, attributes are selected based on the maximum performance of cross-validation. In cross-validation, the structure of a decision table does not change if some instances are inserted or deleted, only the class is changed that is associated with elements (Hall and Frank, 2008).

- **Naive Bayes.** The Naive Bayes approach is a simple probabilistic classifier applied to a classification task (Sahami et al., 1998). It uses probability based on Bayes Theorem, and inputs a node F_i for each of the feature and a class variable C. It assumes all attributes to be independent given a value of a class variable.

- **AdaBoost.** AdaBoost (Freund and Schapire, 1996) is a boosting approach that can reduce an error of any learning approach that generates classifiers whose performance is better compared to random guessing. It constructs an initial

classifier from the original dataset where every sample has an equal distribution ratio of 1. The performance of boosting is better than bagging, and boosting can be used with very simple rules to construct classifiers that are similar to the C4.5 algorithm.

4.5 Experimental Design and Evaluation Results

In this section, I describe datasets and evaluation metrics. I also describe experimental procedures and evaluation results of the terminological matching and combination approaches.

4.5.1 Datasets

Five XDR schemas: CIDX, EXCEL, NORIS, PARAGON and APERTUM of the purchase order domain obtained from www.biztalk.org are used for this evaluation. The schemas CIDX, EXCEL, NORIS, PARAGON and APERTUM are denoted by C, E, N, P, and A respectively. These schemas are used for schema mapping evaluation (Peukert et al., 2011). Table 4.1 summarizes the characteristics of the schemas.

Table 4.1: Characteristics of test schemas

Schema	**Schema Elements**	**Schema Paths**	**Max Path Length**
CIDX (C)	30	34	4
EXCEL (E)	35	48	4
NORIS (N)	46	65	4
PARAGON (P)	82	77	6
APERTUM (A)	59	136	5

In Table 4.1, the schemas contain 30 (C), 35 (E), 46 (N), 82 (A), 59 (P) schema elements and 34 (C), 48 (E), 65 (N), 77 (P) and 136 (A) schema paths. The maximum path lengths of C, E, N, P and A are 4, 4, 4, 6 and 5 respectively. The characteristics of the schemas are described in Figure 4.3 (Madhavan et al., 2001):

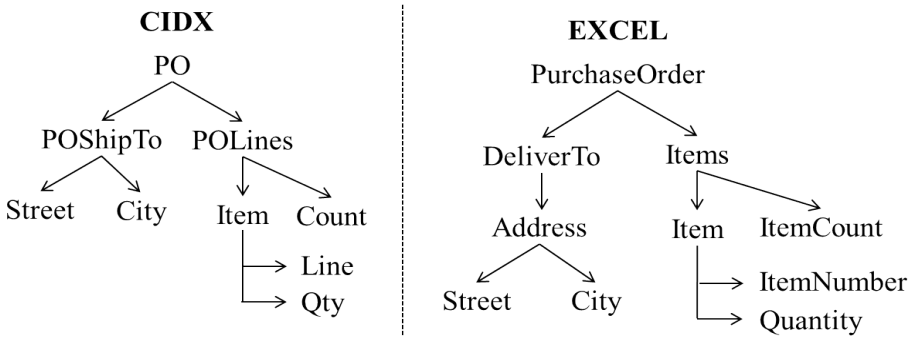

Figure 4.3: Example of schemas

According to the figure, examples of schema elements, schema paths and maximum path lengths are described below:

- Examples of schema elements of CIDX are *PO*, *POShipTo*, *City*, *Line* and *Qty*.
- Examples of schema elements of EXCEL are *PurchaseOrder*, *DeliverTo*, *Address* and *ItemNumber*.
- Examples of schema paths of CIDX and EXCEL are *PO.POLines.Item.Qty* and *PurchaseOrder.Items.Item.Quantity* respectively.
- The maximum paths of CIDX and EXCEL are 4 and 4 respectively.

4.5.2 Evaluation Metrics

In order to evaluate the quality of the automatic match approaches, the first task is to perform matches manually. The real matches (TRUE/FALSE) created manually to produce expert mappings are used as the gold standard (Do et al., 2003). The real matches are later compared to the derived matches found from the automatic match strategies. The comparison is shown in Figure 4.4 (Do et al., 2003).

The set of derived matches is comprised of True Positives (B), False Positives (C) and False Negatives (A) and True Negatives (D). The union of all sets A, B, C, D represents the cross-product of the element sets of two input schemas. I denote True Positive (hit), False Positive (false alarm, Type I error), False Negative (miss, Type II error), and True Negative by TP, FP, FN, and TN respectively. TP means real match

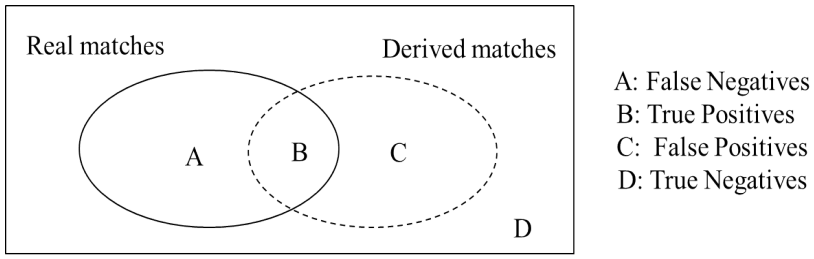

Figure 4.4: Comparison of real matches and derived matches

is true and derived match is true, FP means real match is false and derived match is true, FN means real match is true and derived match is false, and TN means real match is false and derived match is false. For a specific threshold value, TP, FP and FN are calculated by comparing real matches with the derived matches returned by matching algorithms according to Jimenez et al. (2009). Several match quality/performance metrics: precision = TP/(TP+FP), recall = TP/(TP+FN) and F-measure = (2*precision*recall)/(precision+recall) have been proposed in COMA (Do et al., 2003). I used these metrics to assess the quality of the result automatically determined by the match approaches.

4.5.3 Experimental Procedure of Terminological Matching Approaches

In this research, I performed experiments on ten matching tasks one-by-one using all combinations of five schema datasets: C-E (first matching task is to deal with two datasets, CIDX and EXCEL), C-N, C-P, C-A, E-N, E-P, E-A, N-P, N-A and P-A.

I took the Cartesian product of the schema datasets for ten matching tasks separately. The sizes of the Cartesian products of the matching tasks are described in Table 4.2. The sizes of the matching tasks are 1050 (C-E), 1380(C-N), 1770(C-P), 2460(C-A), 1610(E-N), 2065(E-P), 2870(E-A), 2714(N-P), 3772(N-A) and 4838(P-A) element pairs. I performed manual matches for each matching task according to gold standard.

In order to analyze the impacts of different parameter settings, I conducted experiments as follows:

Table 4.2: The sizes of matching tasks

Matching task	**Elements**
EXCEL-CIDX (E-C)	1050
EXCEL-NORIS (E-N)	1610
EXCEL-PARAGON (E-P)	2065
EXCEL-APERTUM (E-A)	2870
CIDX-APERTUM (C-A)	2460
CIDX-NORIS (C-N)	1380
CIDX-PARAGON (C-P)	1770
APERTUM-PARAGON (A-P)	3772
APERTUM-NORIS (A-N)	4838
NORIS-PARAGON (N-P)	2714

- Step 1: Terminological matching performance was compared with different thresholds. The string similarity metrics used in this research are described in Chapter 3. I used string metrics developed by two open source projects. For Levenshtein, JaroWinkler, JaroMeasure, TFIDF and Jaccard, I used open source library SecondString[1] and for MongeElkan, SmithWaterman, NeedlemanWunsch, Q-gram and Cosine, I used SimMetric[2] open source library. Similarity values were normalised such that the values are between 0 to 1, where 0 means strong dissimilarity and 1 means strong similarity. The threshold values for deciding schema matching (TRUE/ FALSE) were increased with 0.1 from 0.1 to 0.9. My main aim in this step was to examine the optimal threshold value for each string metric. I conducted ten matching tasks in this step. After conducting Step 1, the best string metric was chosen for the Step 2 experiment.

- Step 2: Terminological matching was evaluated with individual text processing techniques. The techniques used in this research are the following:

  - **Tokenization processing:** The whole schema elements of two different schema datasets may not be matched, but their tokens may be similar (Ngo

[1] http://secondstring.sourceforge.net

[2] http://sourceforge.net/projects/simmetrics

et al., 2011a). For this reason, I used tokenization for matching schema element *POShipTo* of one schema dataset and schema element *Ship* of another schema dataset. I parsed names of elements into tokens by a customizable tokenizer using punctuation, uppercase, special symbols and digits. *POShipTo* was tokenized into $\{PO, Ship, To\}$. Then *Ship* of one schema dataset was matched to *Ship* of another schema dataset.

- **Abbreviation and Acronym Expansion:** A name of an element may be an abbreviation (*tel* instead of *telephone*) or an acronym (UOM instead of $Unit\ Of\ Measure$). For matching abbreviation and acronym, external resources are required. For this, I used the abbreviation file created for COMA (Do and Rahm, 2002).
- **Synonym processing:** Names of elements of one schema dataset may be synonyms of names of elements of another dataset. I applied synonym matching to use semantically identical schema element names to measure similarity (e.g., element name $Invoice$ is semantically same as element name $Bill$ in the purchase order domain). I used the synonym file created for COMA (Do and Rahm, 2002) as an external dictionary for synonym matching.

- Step 3: The performance of terminological matching can be improved by combining different individual text processing techniques. The combinations are described below:

  - **Tokenization+Abbreviation Matching:** Tokenization is used with abbreviation. An example is $PartNumber \rightarrow PartNO$ where both are tokenized as $\{Part, Number\}$ and $\{Part, NO\}$ respectively. Following this, the abbreviated form NO is expanded as $\{Number\}$. Finally, $PartNumber$ and $PartNO$ are matched by matching $Part$ to $Part$ and $Number$ to $Number$.
  - **Tokenization+Synonym Matching:** Tokenization is also used with synonym. An example of this matching is $Items \rightarrow POLines$. First $POLines$ is tokenized as $\{PO, Lines\}$. Later $Items$ and $Lines$ are matched by the synonym matching process.
  - **Abbreviation+Synonym matching:** Abbreviation and acronym expansion are also necessary for synonym matching. For example, $Qty \rightarrow Amount$.

First, *Qty* is expanded to *Quantity*, and synonym matching is then done between *Quantity* and *Amount*.

- **Tokenization+Abbreviation+Synonym matching:** All the text processing techniques are also used to match names of elements. For example, $POShipTo \rightarrow PurchaseOrderDeliverTo$. In this case, both are tokenized as $\{PO, Ship, To\}$ and $\{Purchase, Order, Deliver, To\}$. PO is then expanded to $\{PurchaseOrder\}$, and later the expanded part is tokenized to $\{Purchase, Order\}$. Finally, synonym matching is done between *Ship* and *Deliver*, and the remaining tokens are matched by string similarity metrics.

I evaluated the terminological matching performance with the above combinations of text processing techniques. The schema datasets used for Step 2 are also used in this step.

4.5.4 Evaluation Results of Terminological Matchers

Choosing a string metric, a threshold value and a text processing technique for matching names of elements depends on characteristics of datasets. The performance of different string metrics can be decreased or increased rapidly according to distinct thresholds. The reason is that distinct metrics have different mathematical properties, and one optimal threshold may not be suitable for all metrics (Stoilos et al., 2005). For each metric and each experiment, it is necessary to change the value of threshold in order to achieve high precision and recall. Precision estimates the reliability of the match predictions, and recall specifies the share of real matches. During schema mapping, manually matching schemas of two heterogeneous data sources and incorrectly identified matches by algorithms are handled by humans. The burden of deleting incorrectly identified matches is much easier than creating manual matches among thousands of schemas (Stoilos et al., 2005). For calculating recall, manually identified matches are required. Recall is therefore very important. Neither precision nor recall alone can estimate the quality of match algorithms (Cheng et al., 2005). So it is important to calculate F-measure of string metrics using both precision and recall. For this, I determined the best performing string metric and threshold value based on the optimized F-measure (Marie and Gal, 2008) for almost all experimental datasets.

i. Schema Mapping Results without Text Processing

The best matching similarity metric based on F-measure without applying any text processing technique for C-E dataset is shown in Figure 4.5.

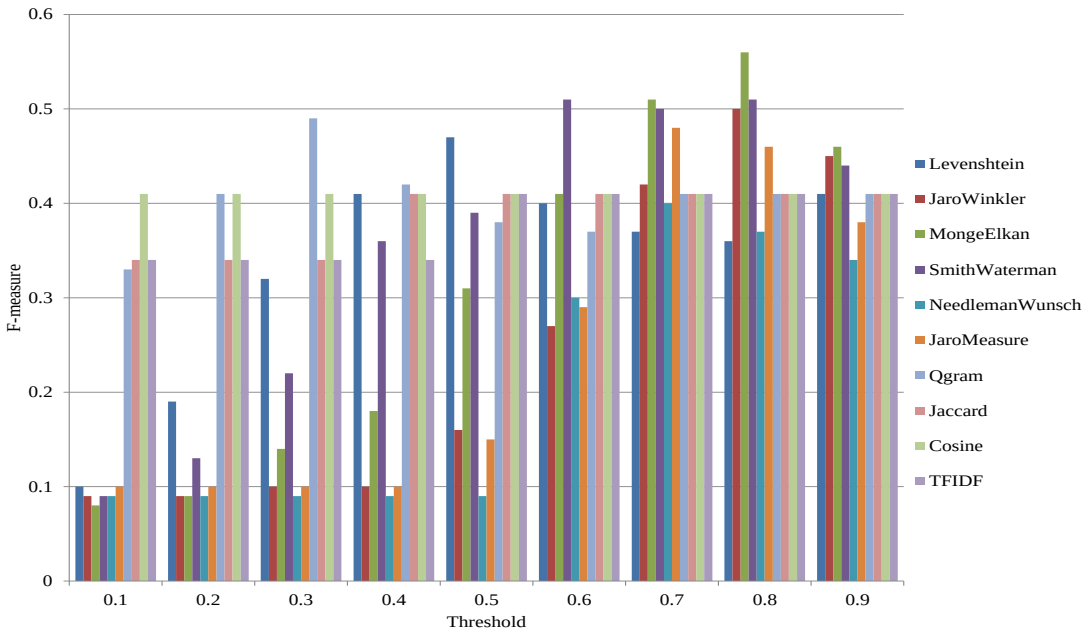

Figure 4.5: Best string metric and threshold without text processing for C-E

The result in Figure 4.5 shows that MongeElkan is the best performing metric at 0.8 threshold value compared to other metrics for C-E dataset. F-measure of the C-E dataset for MongeElkan is 0.56 when no text processing technique is applied. The best matching similarity string metrics based on F-measure without applying any text processing technique for all datasets are summarized in Table 4.3.

In the table, the character-based metrics Jaro, JaroWinkler, SmithWaterman, NeedlemanWunsch and the hybrid metric MongeElkan are the best performing metrics. This is reasonable because an element name usually consists of a string of a word or a string of any number of conjoined words without space, and in this case, the token-based metrics are only effective when element names of a source schema and a target schema are identical. Threshold values for the best performed metrics are slightly different between different datasets, ranging from 0.7 to 0.9. Even though Table 4.3 summaries similarity metrics that performed best, their performance is very low - while precision

Table 4.3: Best mapping results without text processing

Datasets	**String Metric**	**Threshold**	**Precision**	**Recall**	**F-measure**
C-E	MongeElkan	0.8	0.78	0.44	0.56
C-N	MongeElkan	0.8	0.86	0.41	0.56
C-P	JaroWinkler	0.9	0.53	0.24	0.33
C-A	JaroMeasure	0.8	0.64	0.34	0.44
E-N	JaroWinkler	0.8	0.86	0.43	0.58
E-P	MongeElkan	0.7	0.60	0.38	0.46
E-A	JaroWinkler	0.9	0.76	0.36	0.49
N-P	SmithWaterman	0.7	0.63	0.44	0.52
N-A	MongeElkan	0.9	0.83	0.49	0.57
P-A	NeedlemanWunsch	0.8	0.84	0.32	0.46

is ranging from 0.6 to 0.86, recall is ranging from 0.24 to 0.49. This implies that the similarity cannot be used as a sole method for schema mapping, and it is necessary to do further processing to improve the match performance.

ii. Schema Mapping Results with Combined Text Processing

Applying all text processing techniques for the C-E dataset, the best matching similarity metric based on F-measure is shown in Figure 4.6.

The result in the figure shows that Levenshtein is the best performing metric at 0.9 threshold value compared to other string metrics for the C-E dataset. F-measure of the C-E dataset for Levenshtein is 0.74 when the text processing techniques are applied. Table 4.4 summarizes the best schema mapping results when three text processing techniques are applied to the datasets.

In Table 4.4, I show that in general, the performance of schema mapping is significantly improved compared to those of the unprocessed datasets shown in Table 4.3. According to precision, recall and F-measure, most schema datasets obtained high performance when applying Levenshtein similarity metric, but N-A and P-A schema datasets exhibit the highest performance with JaroMeasure and N-P dataset exhibiting the highest performance with NeedlemanWunsch. This implies that it is necesssary

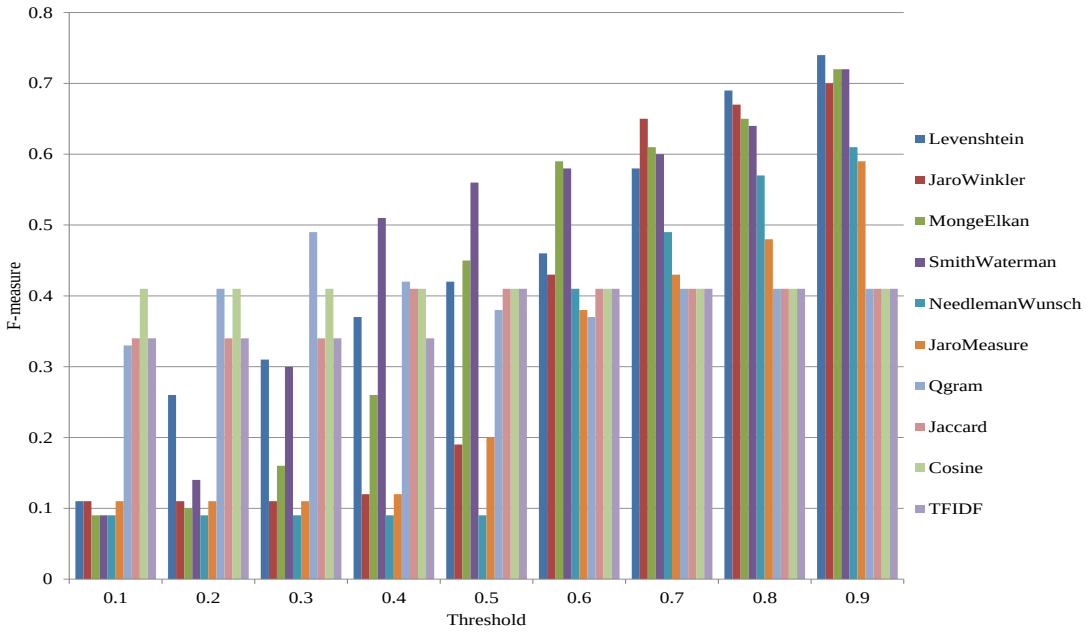

Figure 4.6: Best string metric and threshold with combined text processing for C-E

Table 4.4: Best mapping results with combined text processing

Datasets	**String Metric**	**Threshold**	**Precision**	**Recall**	**F-measure**
C-E	Levenshtein	0.9	0.79	0.70	0.74
C-N	Levenshtein	0.8	0.76	0.74	0.75
C-P	Levenshtein	0.9	0.75	0.71	0.73
C-A	Levenshtein	0.9	0.80	0.78	0.79
E-N	Levenshtein	0.8	0.79	0.74	0.76
E-P	Levenshtein	0.9	0.81	0.75	0.78
E-A	Levenshtein	0.8	0.77	0.82	0.79
N-P	NeedlemanWunsch	0.8	0.84	0.69	0.76
N-A	JaroMeasure	0.9	0.68	0.79	0.73
P-A	JaroMeasure	0.9	0.64	0.68	0.66

to choose appropriate similarity metrics in order to obtain optimal performance when applying similarity metrics with the combined text processing strategies (see Section 4.5.3 Step 3). The results also describe the limits of the similarity metrics based schema mapping. Even though combined application of three text processing techniques improves performance, the schema datasets still exhibit low performance. F-measure of C-E, C-N, C-P, C-A, E-N, E-P, E-A, N-P, N-A, P-A schema datasets are only 0.74, 0.75, 0.73, 0.79, 0.79, 0.76, 0.78, 0.79, 0.76, 0.73 and 0.66 respectively. This implies that additional processing is required to improve performance other than string similarity metrics and text processing techniques.

iii. Performance Improvement Factors

Text processing techniques contribute to performance improvements differently. Figure 4.7 shows how different combinations of text processing techniques with string metric improve F-measure for all datasets. In this figure, SM, $SM + AB$, $SM + TK$, $SM + SN$, $SM + AB + SN$, $SM + TK + AB$, $SM + TK + SN$, $SM + TK + AB + SN$ means string-metric, string-metric+abbreviation, string- metric+tokenization, string-metric+synonym, string-metric+abbreviation+synonym, string-metric+tokenization+ synonym, string-metric+tokenization+abbreviation+synonym respectively.

Figure 4.7 shows that using only string metrics generates low F-measure for all datasets. When text processing techniques are applied, F-measure is then increased gradually. In this figure, string metrics and threshold values are the best performing metrics and threshold values respectively for all the schema datasets described in Table 4.4. It is found that when SM+AB is applied, F-measure is improved compared to only SM for almost all datasets except the E-A dataset. The reason is that the dataset E-A does not contain any abbreviated word. When SM+TK is applied, F-measure is improved significantly for all datasets compared to only SM. Application of SM+SN improves F-measure for all schema datasets except C-A. Then comparing F-measure of SM+AB, SM+TK and SM+SN, I find that tokenization is the highest performing text processing technique compared to others. In the same way, I applied other combinations of string metrics and text processing techniques and display F-measure improvements for all the schema datasets. In this figure, I only show F-measure improvements for the best performing string metrics and threshold values. In the same way, I also obtained the improvements of precision and recall. I display the results in the following three

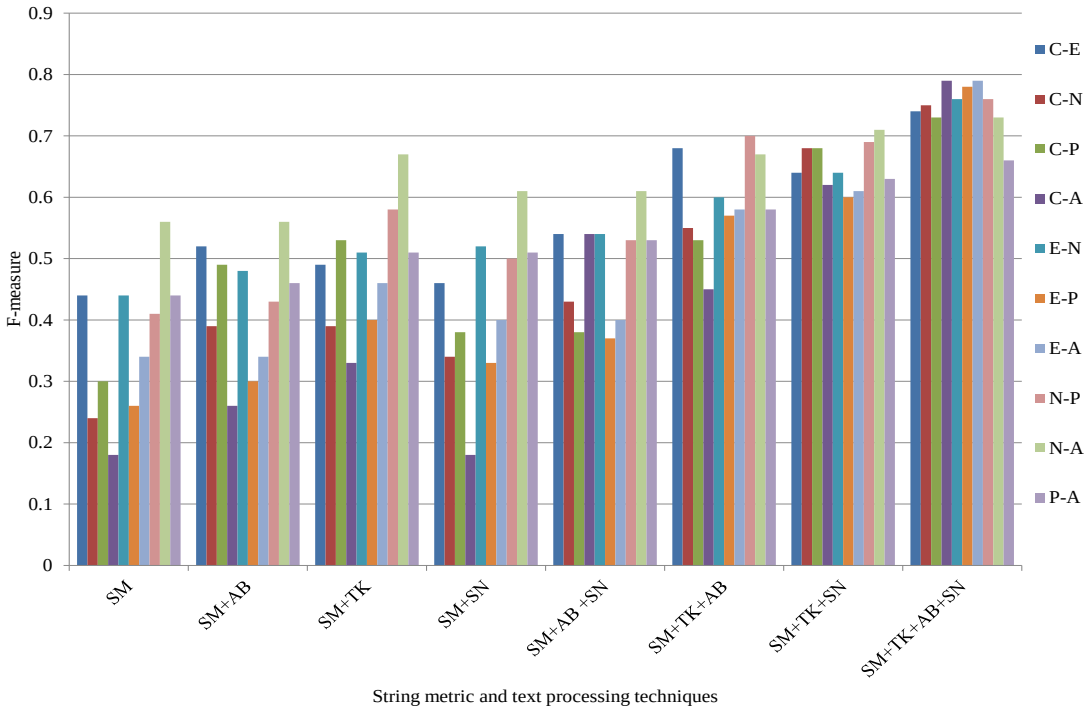

Figure 4.7: String metric and different combinations of text processing techniques

tables.

Tables 4.5, 4.6, 4.7 summarize how precision, recall, F-measures are improved by applying different text processing techniques for the best performed settings that were summarized in Table 4.4. In Tables 4.5, 4.6, 4.7, SM, AB, TK and SN means String Metric, Abbreviation, Tokenization and Synonym respectively, and ALL means calculation similarity values of string metrics with all text processing techniques together. In all the tables, Imp means improvement that is calculated by deducting the value of SM from the value of ALL (ALL - SM). From the analysis of the results of Tables 4.5, 4.6, 4.7, I determine that if the schema datasets contain a large number of combined words (where it is necessary to separate words based on punctuation, uppercase, special symbols, digits and underscores), tokenization performs better compared to others. Abbreviation expansion is useful when schema datasets contain many abbreviated words. If schema datasets contain numerous synonym words, then the performance of synonym matching is better compared to other text processing techniques.

Table 4.5: Performance improvement factors (precision)

Datasets	**SM**	**AB**	**TK**	**SN**	**AB+SN**	**TK+AB**	**TK+SN**	**ALL**	**Imp**
C-E	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.79	-0.21
C-N	1.00	1.00	1.00	1.00	1.00	1.00	0.89	0.76	-0.24
C-P	1.00	1.00	1.00	1.00	1.00	0.94	0.75	0.75	-0.25
C-A	1.00	1.00	1.00	1.00	1.00	1.00	0.73	0.80	-0.20
E-N	1.00	1.00	1.00	1.00	1.00	1.00	0.79	0.79	-0.21
E-P	1.00	1.00	1.00	1.00	1.00	1.00	0.74	0.81	-0.19
E-A	1.00	1.00	0.82	1.00	1.00	0.86	0.69	0.77	-0.23
N-P	1.00	1.00	1.00	1.00	1.00	1.00	0.88	0.84	-0.16
N-A	1.00	1.00	1.00	1.00	1.00	0.83	0.69	0.68	-0.32
P-A	0.83	0.84	0.79	0.86	0.86	0.77	0.65	0.64	-0.19
Avg	0.98	0.98	0.96	0.99	0.99	0.94	0.76	0.76	-0.22

The results show that precision is not improved by using text processing techniques as illustrated in Table 4.5, while recall is significantly improved by using them shown in Table 4.6. The reason for not improving the precision value is that if false positive is 0 (returned by string similarity metric), precision is 1.0 because precision is calculated

by true positive/(true positive+ false positive). The tokenization processing generally gives the most significant contribution to performance improvement (see Table 4.6 and 4.7). On average, the tokenization technique increases recall and F-measure up to 0.34 and 0.49 respectively, while the abbreviation technique increases up to 0.27 and 0.42 respectively and the synonym increases up to 0.28 and 0.42 respectively.

Table 4.6: Performance improvement factors (recall)

Datasets	**SM**	**AB**	**TK**	**SN**	**AB+SN**	**TK+AB**	**TK+SN**	**ALL**	**Imp**
C-E	0.28	0.35	0.33	0.30	0.37	0.51	0.56	0.70	0.42
C-N	0.17	0.24	0.24	0.21	0.28	0.38	0.55	0.74	0.57
C-P	0.18	0.32	0.37	0.24	0.24	0.37	0.62	0.71	0.53
C-A	0.10	0.15	0.20	0.10	0.37	0.29	0.54	0.78	0.68
E-N	0.29	0.31	0.34	0.35	0.37	0.43	0.54	0.74	0.45
E-P	0.15	0.18	0.25	0.20	0.22	0.40	0.50	0.75	0.60
E-A	0.20	0.20	0.32	0.25	0.25	0.43	0.55	0.82	0.62
N-P	0.26	0.28	0.41	0.33	0.36	0.54	0.56	0.69	0.43
N-A	0.39	0.39	0.58	0.44	0.44	0.56	0.74	0.79	0.40
P-A	0.30	0.32	0.38	0.36	0.38	0.46	0.60	0.68	0.38
Avg	0.23	0.27	0.34	0.28	0.33	0.45	0.58	0.74	0.51

In some cases such as E-N dataset, the synonym technique improves F-measure better than the tokenization technique. The reason is that the number of synonym words is larger than combined words in this dataset. When the two measures are combined, the combination of the tokenization and synonym techniques increases F-measure most significantly (on average up to 0.65). This is due to the fact that datasets contain more combined words where tokenization is required to split the words, and later synonym look up is required to match. The combination of the tokenization and the abbreviation techniques improves F-measure most in the N-P dataset. When the three text processing techniques are combined together, all datasets exhibit the highest performance. However, there are several datasets that exhibit very similar recall and F-measure when only two techniques are combined together.

Table 4.7: Performance improvement factors (F-measure)

Datasets	**SM**	**AB**	**TK**	**SN**	**AB+SN**	**TK+AB**	**TK+SN**	**ALL**	**Imp**
C-E	0.44	0.52	0.49	0.46	0.54	0.68	0.64	0.74	0.30
C-N	0.24	0.39	0.39	0.34	0.43	0.55	0.68	0.75	0.51
C-P	0.30	0.49	0.53	0.38	0.38	0.53	0.68	0.73	0.43
C-A	0.18	0.26	0.33	0.18	0.54	0.45	0.62	0.79	0.61
E-N	0.44	0.48	0.51	0.52	0.54	0.60	0.64	0.76	0.32
E-P	0.26	0.30	0.40	0.33	0.37	0.57	0.60	0.78	0.52
E-A	0.34	0.34	0.46	0.40	0.40	0.58	0.61	0.79	0.45
N-P	0.41	0.43	0.58	0.50	0.53	0.70	0.69	0.76	0.35
N-A	0.56	0.56	0.67	0.61	0.61	0.67	0.71	0.73	0.17
P-A	0.44	0.46	0.51	0.51	0.53	0.58	0.63	0.66	0.22
Avg	0.36	0.42	0.49	0.42	0.49	0.57	0.65	0.75	0.39

4.5.5 Summary of Terminological Schema Matching

The performance of terminological matching using text processing techniques with different threshold values is described above. In order to achieve high match accuracy, it is important to combine different string metrics with distinct text processing. Using one fixed combination does not work well for matching all element pairs because elements contain different characteristics. In this research, I evaluated various string similarity metrics with different thresholds and with various combinations of text processing techniques. Though recall and F-measure are increased for all datasets, they are still not considered to be high enough. In order to improve match performance, it is necessary to handle schema matching problems of false positives and false negatives using combination approaches - machine learning and knowledge engineering. In the next section, the combination approaches are used to perform element-level matching.

4.5.6 Experimental Procedure of Combination Approaches

In this research, I performed experiments on ten matching tasks. The matching tasks C-E, C-N, C-P, C-A, E-N, E-P, E-A, N-P, N-A and P-A are described in Section 4.5.3. I denote the matching tasks by D1, D2, D3, D4, D5, D6, D7, D8, D9 and D10

respectively. In order to use the datasets for classification and to provide appropriate knowledge to the users for knowledge acquisition, feature construction is required.

i. Feature Construction

Features are constructed based on input schemas, application of string similarity metrics and string text processing techniques on the schemas, and gold standard. The string similarity metrics and text processing techniques are described in Chapter 3. Feature construction is described in Figure 4.8. The following steps are used for constructing features and feature values for each pair of elements:

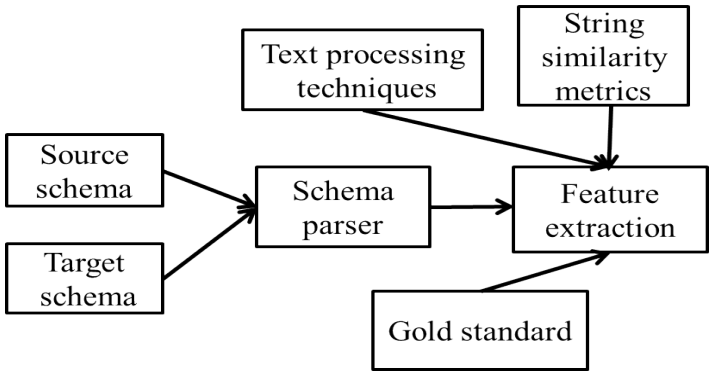

Figure 4.8: Feature construction

- **Step 1 Inputs:** In this step, names of elements of a source schema and a target schema are used as input. For example, *Item* and *Line* are source and target element names respectively.
- **Step 2 Application of string text processing techniques:** In this step, different combinations of string text processing techniques such as tokenization, abbreviation and acronym expansion and synonym matching are applied to the input schemas. For example, applying synonym matching to the target element name *Line*, *Item* is obtained. This is because *Line* and *Item* are synonyms in the purchase order domain.
- **Step 3 Application of string similarity metrics:** In this step, string similarity metrics are applied on the features computed from steps 1 and 2. String

similarity metrics developed by two open source projects are used. For Levenshtein, JaroWinkler, JaroMeasure, TFIDF and Jaccard, open source library SecondString[3] is used and for MongeElkan, SmithWaterman, NeedlemanWunsch, Q-gram and Cosine, SimMetric[4] open source library is used. Similarity values are normalized, such that the values are from 0 to 1, where 0 means strong dissimilarity and 1 means strong similarity. For example, if Levenshtein function is applied to the names of schema elements *Items* and *Item*, the similarity value 0.8 is obtained. The threshold values are increased with 0.1 from 0 to 1.

- **Step 4 Using gold standard:** The gold standard values (TRUE/ FALSE) provided manually by experts between pairs of elements, are also used. The values are termed *Class* which creates another feature.

By applying the four steps, 73 features are obtained by using schema information of each matching task. Constructed features represent knowledge about relationships among attributes, operators or process patterns. The prepared features and feature values of the features of each matching task are termed *attributes* and *cases* respectively. These are used later in the CPR based RDR and machine learning approaches.

ii. Evaluation of CPR based RDR and Machine Learning Approaches

Each dataset was processed using the CPR based RDR and machine learning approaches. These approaches learn a new model using newly available data. In the evaluation, I performed ten experiments to obtain the precision, recall and F-measure of CPR based RDR and machine learning approaches. In all experiments, I randomly selected datasets for training and testing. For example, $D1$ was selected for training and $D10$ for testing, $D7$ for training and $D3$ for testing and so on. The evaluation of CPR based RDR and machine learning approaches are described below:

A. Evaluation of CPR based RDR

The evaluation of CPR based RDR is described in Figure 4.9:

[3] http://secondstring.sourceforge.net

[4] http://sourceforge.net/projects/simmetrics

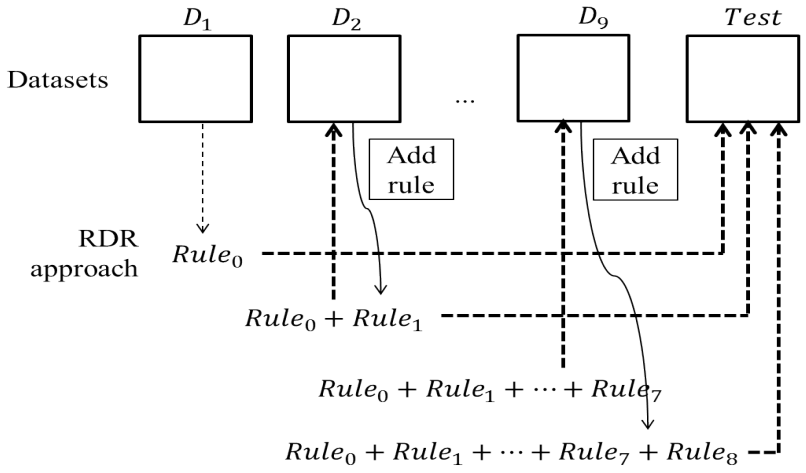

Figure 4.9: Evaluation approach of the CPR based RDR

In the CPR based RDR approach, all the cases (pairs of elements) of one dataset are matched by creating rules. Later, the rules are used to predict the matching of the unseen pairs of elements of the test dataset. In this way, the matching task is transformed into a classification task (Ngo et al., 2013). First, one dataset is classified by creating rules $Rule_0$. These rules are used to test the test dataset. The rules are also used to test another dataset. If the rules find some incorrectly classified cases, then these cases are refined by censor rules $Rule_1$ to make the classification NULL. Following this, the NULL classified cases are classified correctly by alternative rules $Rule_1$. This rules make the cases TRUE/FALSE classified according to gold standard. Here, the names of the censor and alternative rules are the same $Rule_1$. The conditions of this rules can be the same or different. The censor rules are added as censor/child nodes of the parent rules and alternative rules are added as child nodes of the root rule to the KB. The $Rule_0$+$Rule_1$ is then used to test the test dataset and also to test another new dataset. If any incorrectly classified case is found, then censor and alternative rules $Rule_2$ are added to correct the classification. Later, $Rule_0$+$Rule_1 + Rule_2$ is used to test the test dataset. In such a way, rules $Rule_0$+$Rule_1 + Rule_2 + ... + Rule_9$ are added for all nine datasets and used to test the test dataset.

B. Evaluation of Machine Learning Approaches

I divide the machine learning approaches into two approaches, static and dynamic. These are described below:

Static Machine Learning: In the static machine learning approach, classification models are created separately for all datasets excluding the test dataset, and the models are used to test the test dataset. For example, a machine learning model ML_0 is created for $D1$ and the model is used to test $D10$. Then ML_1 is created for $D2$ and this ML_1 is used to test $D10$. In this way, ML_2 for $D3$ to ML_8 for $D9$ are created. For other combinations, ML_0 is created for $D7$ and used to test $D3$, ML_1 is created for $D8$ and used to test $D3$. In this way, ML_8 is created for $D1$ and used to test $D3$.

Dynamic Machine Learning: The evaluation of dynamic machine learning approach is described in Figure 4.10:

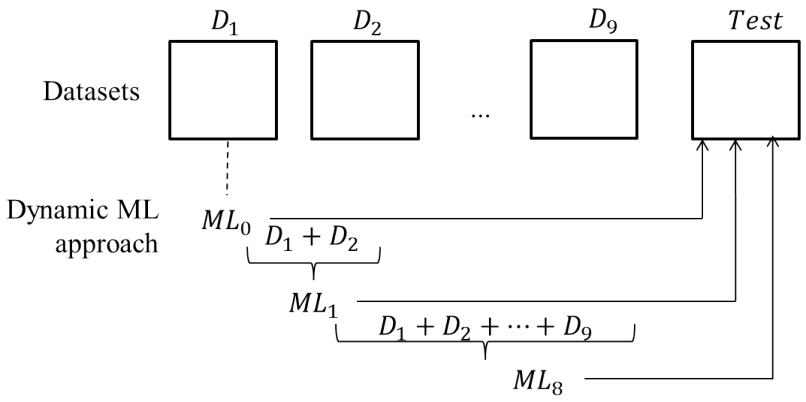

Figure 4.10: Dynamic machine learning approach

In the dynamic machine learning approach, a classification model is created for one dataset and the model is used to test the test dataset. Other datasets are added incrementally to create models, and the models are used separately to test the test dataset. For example, a model, ML_0 is created for $D1$ and used to test $D10$. Following this, other datasets are incrementally added like $D1 + D2$, $D1 + D2 + D3$ to create models ML_1, ML_2 respectively, and the models are used separately to test $D10$. In this way, all nine datasets are added to create a model ML_8 and this model is used to test $D10$.

iii. Schema Mapping by CPR based RDR

I developed a schema matching system with a simple GUI for schema mapping using the CPR based RDR approach. The system can select any dataset from a repository. Features created by the *Feature Construction* processes (Section 4.5.6(i)) are represented in a *Case Browser* to provide sufficient knowledge to the users (Figure 4.11). The features and feature values are called attributes and cases respectively. The system works in two phases: a learning phase and a testing phase. In the learning phase, the *Add Classification* button of Figure 4.11 is used. In the testing phase, the *Classify* button of this figure is used.

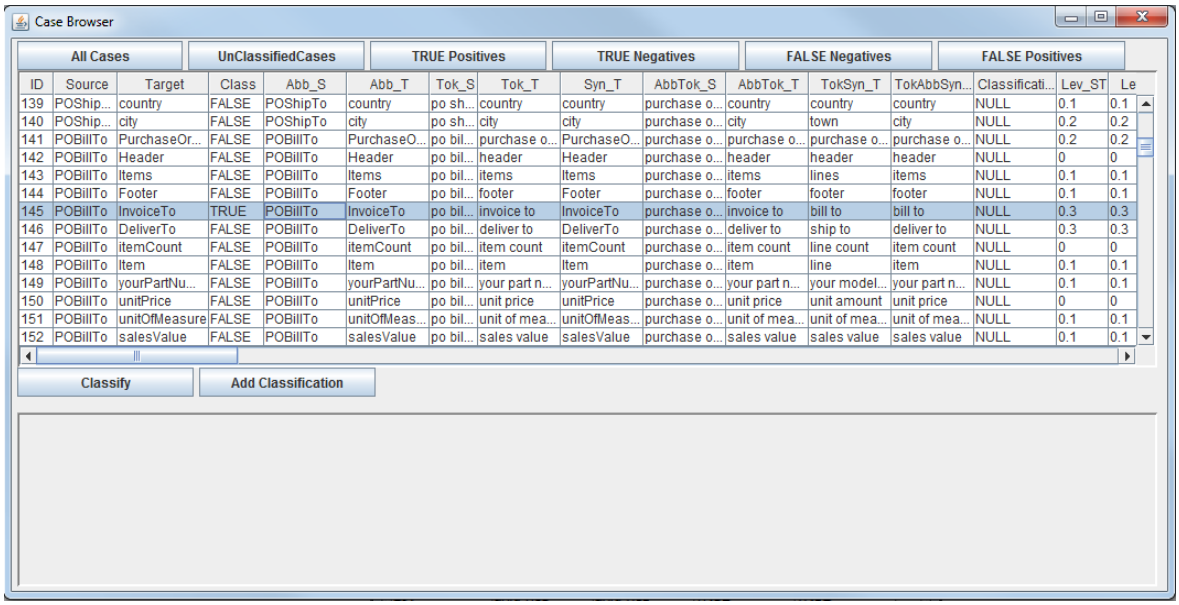

Figure 4.11: GUI represents 73 features with feature values

In Figure 4.11, the *Add Classification* button adds a classification for a selected case. To add a classification, the knowledge acquisition GUI is displayed in Figure 4.12. In this figure, classification of the rule is first selected. This can be done using the drop down box at the top which lists TRUE or FALSE classification for this domain. Having selected the classification, conditions of a rule are added. For each condition, *attribute*, *operator* and *value* are selected from the drop down boxes consisting of a list of attributes, operators and values. After selecting a condition, the *Add Condition* button adds the condition. It is possible to add more than one condition by the same way. The *Delete Selected* button is used to delete a condition if users think

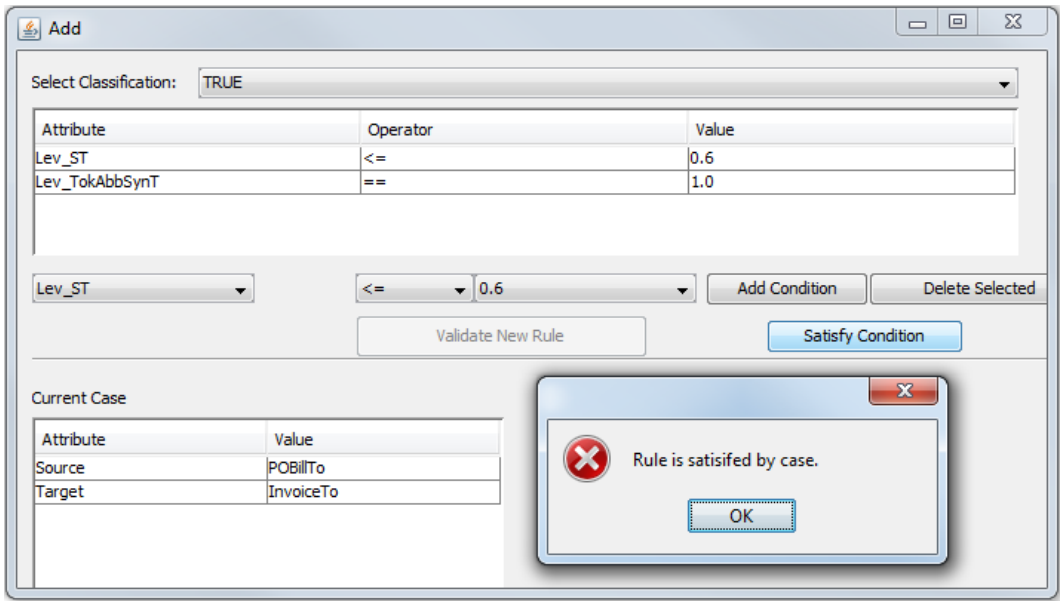

Figure 4.12: Knowledge acquisition GUI for adding rules

that the added condition is not suitable. The *Satisfy Condition* button helps to look at whether the rule is satisfied by the selected case or not. If it is satisfied, the *Validate New Rule* button then becomes active, and this helps to validate the rule on unclassified cases of datasets (see Figure 4.11).

In Figure 4.13, the *Reported Match* shows the manual matching results, and the *Algorithmic Match* shows the results calculated by the rule. The *Save Rule* button helps to save the *rule* in the rule database KB and the *case* in the case database. The *Edit Classification* button helps to refine the incorrectly classified cases by adding new conditions or creating another new rule until all incorrect cases are removed using the knowledge acquisition GUI. Knowledge acquisition for editing classification is shown in Figure 4.14.

Classification of a censor rule is always NULL. The refined cases and the deleted incorrectly classified cases from the satisfied cases list are shown in Figure 4.15. Here the *Save Rule* button saves a censor rule in the rule database, KB, as a censor/child node of a parent rule and the deleted cases in the case database as the NULL classification. If there are more incorrectly classified cases, the rule can be refined by adding other censor rules. NULL classified cases are later classified by alternative rules created by the *Add Classification* button (see Figure 4.11). The alternative rules are added as

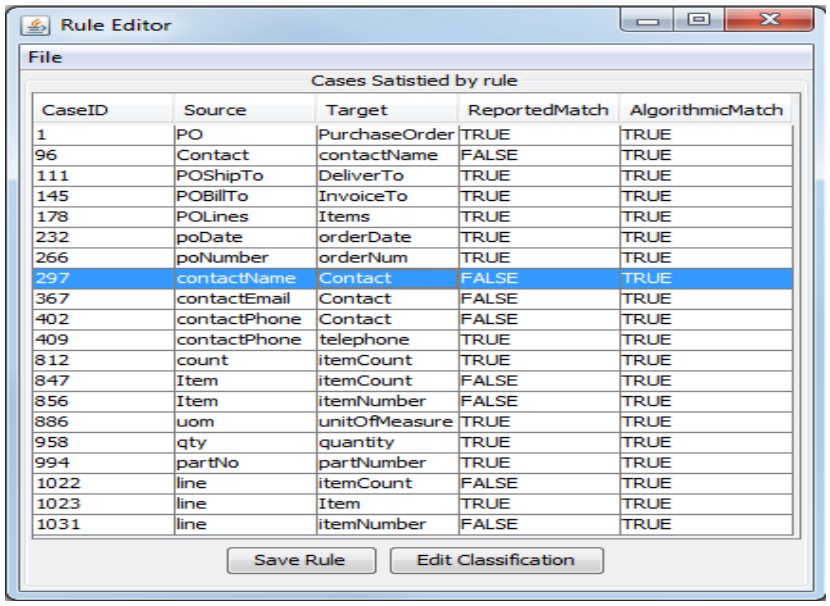

Figure 4.13: Satisfied cases

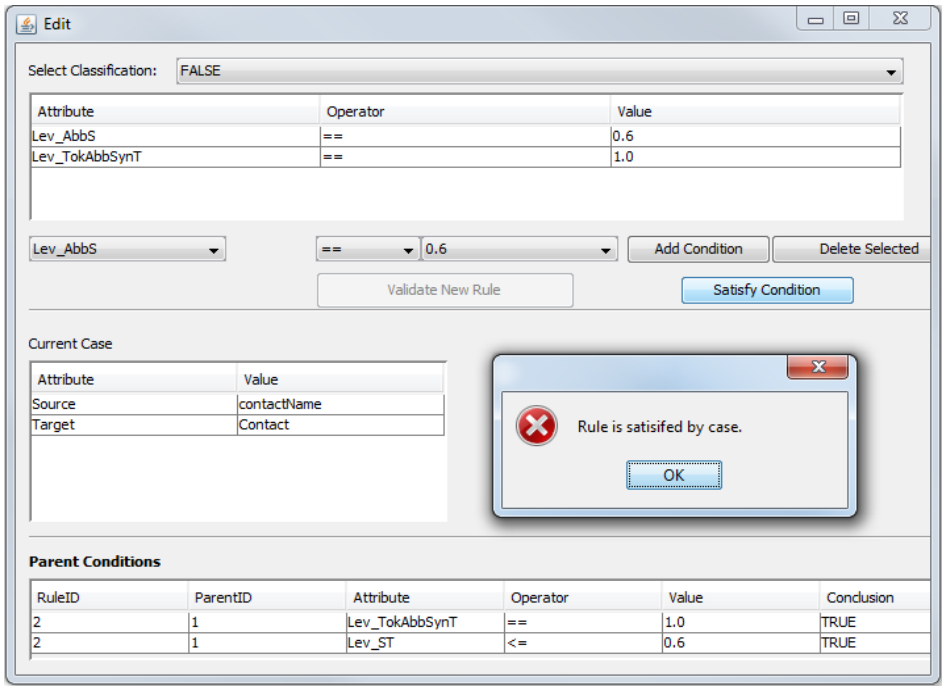

Figure 4.14: Knowledge acquisition for editing classification

parent rules to the KB. These parent rules are also called child rules of the root rule of the KB.

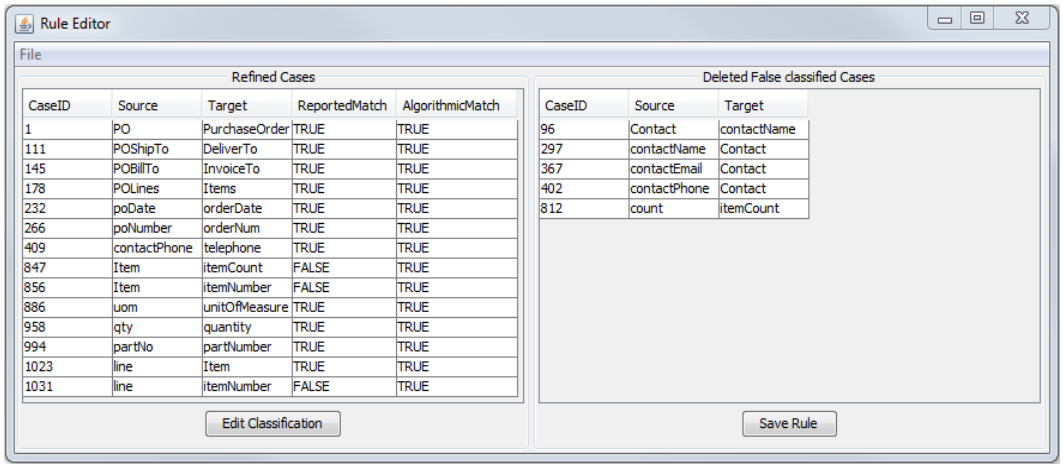

Figure 4.15: Deleted incorrectly classified cases

In the testing phase, the *Classify* button of Figure 4.11 classifies test cases using the rules created by the above procedure.

iv. Schema Mapping by Machine Learning Approaches

In order to determine similarities between two datasets, one dataset is trained using J48, RandomForest, REPTree, BFTree, ADTree, Functional Tree, SimpleCart, NBTree, DecisionTable, Naive Bayes and AdaBoost1. The features created by the *Feature Construction* processes are used as training samples to build a model using one approach separately. The purpose of building a model is to classify whether element pairs of schema datasets are matched or not based on their feature similarity measures. For all machine learning approaches, 10-fold cross validation is used. This means data is split into 10 groups where nine groups are considered for training and the remaining one group is considered for testing. This process is repeated for all 10 groups. Then average precision, recall and F-measure of all experiments are calculated and they are compared among the machine learning approaches.

4.5.7 Evaluation Results of CPR based RDR and Machine Learning Approaches

The match performance of the rule-based system and machine learning approaches depends on the characteristics of the datasets. The performance of the rule-based system also depends on efficient rule creation. For example, when datasets are used to train by the machine learning approaches and to create rules by CPR based RDR contain a large number of combined words, but the test datasets do not contain a large number of combined words, performance then becomes low. I determined the best performing classification system based on the optimized F-measure (Marie and Gal, 2008) for almost all experimental datasets. For all experiments using the machine learning approaches, I used the WEKA (Hall et al., 2009) data mining and machine learning toolbox. For all experimental results, I denote Datasets, Performance Metric, CPR based RDR rule-based schema matching system, J48, RandomForest, REPTree, BFTree, ADTree, Functional Tree, SimpleCart, NBTree, DecisionTable, Naive Bayes, AdaBoost1 and average by D, PM, RDR, J48, RF, RP, BT, AT, FT, SC, NT, DT, NB, AB and Avg respectively.

i. Performance Comparison among CPR based RDR and Machine Learning approaches

The performance measures of CPR based RDR and machine learning approaches are described below. I then evaluated the results to compare the performance of the approaches.

A. Schema Mapping Results with One Training dataset

I used one dataset for creating rules using CPR based RDR. I first created on average 25 rules to classify the cases. The detail rule creation process is explained in Section 4.5.6. I also used the same dataset to feed into machine learning approaches for training. I used other datasets to test. The performance of the rule-based system and machine learning approaches is summarized in Table 4.8.

The results in Table 4.8 indicate that using CPR based RDR, the performance is high for $D1$, $D2$, $D4$, $D5$, $D6$, $D7$, and $D8$ datasets in terms of precision, and it is also

Table 4.8: Performance comparison with one dataset for training

D	PM	RDR	J48	RF	RP	BT	AT	FT	SC	NT	DT	NB	AB
	P	0.92	0.75	0.87	0.73	0.89	0.85	0.82	0.89	0.71	0.84	0.39	0.91
D1	R	0.73	0.60	0.63	0.38	0.65	0.59	0.41	0.65	0.74	0.57	0.77	0.48
	F	0.79	0.66	0.73	0.50	0.75	0.69	0.53	0.75	0.73	0.68	0.51	0.62
	P	0.94	0.78	0.78	0.54	0.78	0.79	0.72	0.76	0.72	0.76	0.33	0.73
D2	R	0.57	0.68	0.77	0.85	0.68	0.68	0.80	0.70	0.81	0.70	0.91	0.70
	F	0.71	0.73	0.77	0.65	0.72	0.73	0.75	0.73	0.76	0.73	0.49	0.71
	P	0.83	0.71	0.86	0.78	0.85	0.69	0.74	0.74	0.73	0.70	0.30	0.77
D3	R	0.64	0.67	0.71	0.64	0.49	0.62	0.74	0.52	0.83	0.44	0.94	0.49
	F	0.72	0.68	0.78	0.70	0.62	0.65	0.74	0.61	0.77	0.53	0.45	0.59
	P	0.92	0.71	0.81	0.56	0.78	0.78	0.76	0.78	0.68	0.79	0.31	0.77
D4	R	0.68	0.75	0.82	0.54	0.71	0.82	0.81	0.71	0.68	0.74	0.94	0.65
	F	0.78	0.73	0.81	0.54	0.74	0.80	0.78	0.74	0.68	0.76	0.46	0.70
	P	0.88	0.48	0.66	0.56	0.53	0.60	0.59	0.53	0.67	0.70	0.32	0.61
D5	R	0.63	0.61	0.69	0.61	0.60	0.65	0.71	0.65	0.77	0.68	0.93	0.64
	F	0.73	0.52	0.67	0.58	0.56	0.62	0.62	0.58	0.72	0.68	0.47	0.62
	P	0.87	0.82	0.86	0.77	0.86	0.79	0.78	0.87	0.72	0.78	0.28	0.80
D6	R	0.69	0.68	0.73	0.77	0.67	0.66	0.73	0.56	0.67	0.67	0.94	0.59
	F	0.77	0.74	0.79	0.77	0.75	0.72	0.75	0.68	0.69	0.72	0.42	0.68
	P	0.91	0.82	0.83	0.77	0.81	0.68	0.57	0.87	0.57	0.83	0.31	0.75
D7	R	0.62	0.63	0.80	0.64	0.58	0.70	0.76	0.56	0.75	0.62	0.94	0.54
	F	0.74	0.70	0.81	0.70	0.68	0.68	0.65	0.68	0.65	0.71	0.46	0.62
	P	0.91	0.69	0.80	0.58	0.84	0.69	0.66	0.84	0.54	0.73	0.30	0.68
D8	R	0.59	0.72	0.73	0.65	0.71	0.62	0.82	0.70	0.60	0.65	0.94	0.55
	F	0.72	0.70	0.76	0.61	0.76	0.64	0.72	0.76	0.57	0.68	0.45	0.60
	P	0.87	0.86	0.89	0.83	0.88	0.84	0.88	0.88	0.60	0.87	0.41	0.65
D9	R	0.64	0.50	0.51	0.43	0.50	0.50	0.54	0.50	0.54	0.48	0.78	0.49
	F	0.74	0.63	0.65	0.56	0.63	0.63	0.66	0.63	0.57	0.61	0.53	0.56
	P	0.85	0.86	0.84	0.90	0.78	0.89	0.79	0.78	0.63	0.82	0.31	0.67
D10	R	0.69	0.56	0.71	0.43	0.63	0.33	0.72	0.63	0.58	0.55	0.94	0.47
	F	0.76	0.68	0.77	0.57	0.69	0.50	0.75	0.69	0.60	0.66	0.46	0.55
	P	**0.89**	**0.75**	**0.82**	**0.70**	**0.80**	**0.76**	**0.73**	**0.79**	**0.66**	**0.78**	**0.33**	**0.73**
Avg	**R**	**0.65**	**0.64**	**0.71**	**0.59**	**0.62**	**0.62**	**0.70**	**0.62**	**0.70**	**0.61**	**0.90**	**0.56**
	F	**0.75**	**0.68**	**0.75**	**0.62**	**0.69**	**0.67**	**0.71**	**0.69**	**0.67**	**0.68**	**0.47**	**0.63**

high for $D1$, $D5$, $D9$ datasets in terms of F-measure. J48 shows the same precision between $D6$ and $D7$ datasets, and Random Forest shows the same F-measure between $D7$ and $D8$ datasets. The performance of Random Forest is high in terms of precision for $D2$, $D3$, $D4$, $D5$, $D6$, $D7$, $D8$ and $D10$ datasets. Though the performance of Naive Bayes is very low according to precision and F-measure, it is higher according to recall compared to other approaches. The reason for high precision means less false positive values, and high recall means that the false negative numbers are very low (Marie and Gal, 2008). The average precision of CPR based RDR approach is higher compared to all the machine learning approaches. The average F-measure of CPR based RDR approach is similar to Random Forest, and average recall is low. To increase precision, recall and F-measure, I performed further processing by adding more datasets incrementally. For the rule-based system, I created rules incrementally for all the datasets excepts the test dataset, and then I tested the test dataset. For the machine learning approaches, I incrementally added datasets for creating models and then tested the test dataset. I do not show the results of all combinations such as two datasets, three datasets here, only the results with the combination of nine datasets are shown in order to describe the incremental performance.

B. Schema Mapping Results with the Combination of Nine Datasets for Training

Table 4.9 summarizes the schema mapping results using the dynamic machine learning approaches. In these approaches, it is necessary to rebuild a training model when datasets are added incrementally. However, for the rule-based system, it is not necessary to create rules for the new dataset from the very beginning. I used the 25 rules created for one dataset in the first experiment in Section 4.5.7 to classify another dataset. I found that some cases are not classified correctly by the rules. I added censor rules to make the classification NULL and added alternative rules to classify the cases correctly. I then reused all the rules to classify another new dataset. When some cases were not classified correctly, I added new rules to correct the classifications. In this way, I classified all the datasets. The classification processes incrementally built the KB. Finally, I used all the rules to test the test dataset to obtain the results.

In Table 4.9, I show that the performance of CPR based RDR and almost all machine learning approaches is improved in terms of precision, recall and F-measure.

Table 4.9: Performance comparison with the combination of nine datasets for training

D	PM	RDR	J48	RF	RP	BT	AT	FT	SC	NT	DT	NB	AB
	P	0.89	0.86	0.90	0.87	0.87	0.85	0.84	0.86	0.83	0.85	0.47	0.89
D1	R	0.84	0.85	0.77	0.93	0.63	0.71	0.92	0.79	0.79	0.82	1.00	0.81
	F	0.86	0.85	0.83	0.90	0.73	0.77	0.87	0.82	0.81	0.83	0.64	0.85
	P	0.91	0.90	0.91	0.86	0.89	0.71	0.91	0.91	0.85	0.86	0.35	0.85
D2	R	0.85	0.84	0.86	0.83	0.75	0.74	0.79	0.69	0.76	0.83	0.95	0.80
	F	0.88	0.87	0.88	0.84	0.82	0.73	0.84	0.78	0.80	0.85	0.51	0.82
	P	0.84	0.83	0.86	0.72	0.86	0.81	0.81	0.77	0.83	0.79	0.34	0.83
D3	R	0.85	0.74	0.82	0.77	0.71	0.71	0.71	0.71	0.71	0.73	0.88	0.79
	F	0.84	0.78	0.84	0.74	0.78	0.75	0.75	0.74	0.76	0.76	0.50	0.81
	P	0.88	0.89	0.92	0.77	0.81	0.90	0.88	0.77	0.86	0.81	0.31	0.80
D4	R	0.83	0.67	0.77	0.69	0.82	0.80	0.62	0.59	0.73	0.74	0.89	0.76
	F	0.85	0.76	0.84	0.73	0.81	0.85	0.72	0.67	0.79	0.78	0.47	0.78
	P	0.85	0.89	0.92	0.83	0.83	0.81	0.87	0.96	0.81	0.77	0.37	0.80
D5	R	0.79	0.77	0.78	0.75	0.71	0.83	0.73	0.69	0.74	0.77	0.88	0.70
	F	0.82	0.82	0.85	0.79	0.76	0.82	0.80	0.80	0.78	0.77	0.51	0.75
	P	0.90	0.90	0.87	0.83	0.86	0.81	0.86	0.82	0.79	0.83	0.38	0.76
D6	R	0.85	0.80	0.80	0.77	0.73	0.75	0.75	0.78	0.75	0.79	0.87	0.76
	F	0.87	0.84	0.83	0.80	0.79	0.78	0.80	0.80	0.77	0.81	0.52	0.76
	P	0.89	0.87	0.89	0.77	0.83	0.82	0.82	0.80	0.88	0.79	0.32	0.79
D7	R	0.82	0.73	0.81	0.62	0.80	0.69	0.76	0.71	0.68	0.84	0.93	0.80
	F	0.85	0.79	0.85	0.69	0.81	0.75	0.78	0.75	0.76	0.81	0.47	0.79
	P	0.88	0.87	0.92	0.75	0.80	0.70	0.85	0.74	0.71	0.79	0.31	0.76
D8	R	0.78	0.68	0.71	0.74	0.73	0.79	0.64	0.90	0.64	0.69	0.89	0.72
	F	0.83	0.76	0.80	0.74	0.76	0.74	0.73	0.81	0.68	0.74	0.46	0.74
	P	0.84	0.71	0.69	0.70	0.72	0.69	0.70	0.76	0.84	0.88	0.28	0.82
D9	R	0.76	0.81	0.75	0.60	0.64	0.71	0.75	0.74	0.59	0.68	0.97	0.77
	F	0.80	0.75	0.72	0.65	0.68	0.70	0.72	0.75	0.69	0.76	0.43	0.79
	P	0.81	0.64	0.65	0.56	0.60	0.62	0.66	0.68	0.56	0.62	0.23	0.61
D10	R	0.73	0.58	0.62	0.54	0.56	0.57	0.56	0.62	0.60	0.61	0.86	0.59
	F	0.77	0.60	0.63	0.55	0.58	0.59	0.58	0.65	0.58	0.61	0.37	0.59
	P	**0.87**	**0.84**	**0.85**	**0.75**	**0.81**	**0.77**	**0.82**	**0.81**	**0.80**	**0.80**	**0.34**	**0.79**
Avg	**R**	**0.81**	**0.75**	**0.77**	**0.72**	**0.71**	**0.73**	**0.72**	**0.72**	**0.70**	**0.75**	**0.91**	**0.75**
	F	**0.84**	**0.79**	**0.80**	**0.74**	**0.75**	**0.75**	**0.76**	**0.76**	**0.74**	**0.77**	**0.49**	**0.77**

However, the performance of Naive Bayes is not increased significantly according to precision and F-measure. Recall of Naive Bayes is always high because its false negative numbers are very low. The average performance of the CPR based RDR approach is the highest in terms of precision and F-measure compared to all the machine learning approaches. The reason is that the CPR based RDR approach can increase precision, recall and F-measure by an incremental knowledge acquisition process.

4.5.8 Prune Tree of J48 and KB of CPR based RDR

Examples of prune trees for training one dataset and the combination of two datasets using J48 are given in Figures 4.16 and 4.17. It is found that the prune tree for training one dataset is different from the prune tree of the combination of two datasets for training.

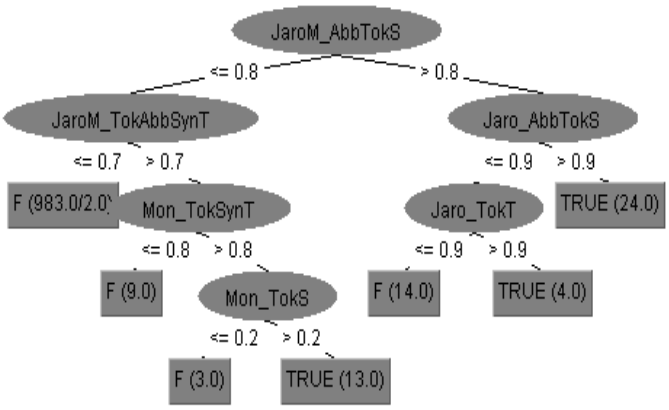

Figure 4.16: J48 prune tree for training one dataset

In Figures 4.16 and 4.17, the abbreviations *JaroM*, *Jaro*, *Mon*, *Lev*, *AbbTokS*, *TokSynT*, *TokAbbSynT*, *TokT* means JaroMeasure, JaroWinkler, MongeElkan, Levenshtein functions, abbreviation and tokenization of source, tokenization and synonym of target, tokenization, abbreviation and synonym of target, and tokenization of target respectively. The values 0.9, 0.8, 0.7, 0.2, 0.1 are thresholds. As an example, the rule Jaro_TokT <= 0.9 means if the value of JaroWinkler function applied to the tokenization of target is less than or equal to the threshold value 0.9, then the conclusion is FALSE.

In the above figures, the TRUE and FALSE conditions are different.

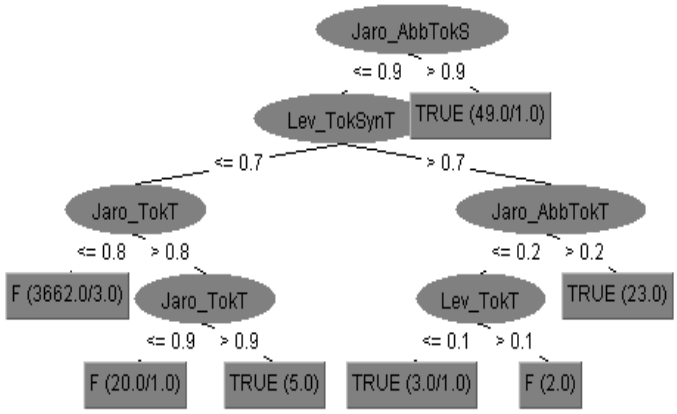

Figure 4.17: J48 prune tree for training the combination of two datasets

In Figure 4.16,

The TRUE conditions are:

- JaroM_AbbTokS <=0.8 and JaroM_TokAbbSynT > 0.7 and Mon_TokSynT > 0.8 and Mon_TokS > 0.2
- JaroM_AbbTokS >0.8 and Jaro_AbbTokS > 0.9
- JaroM_AbbTokS >0.8 and Jaro_AbbTokS <= 0.9 and Jaro_TokT > 0.9

For example, if JaroM_AbbTokS >0.8 and Jaro_AbbTokS > 0.9 then TRUE.

The FALSE conditions are:

- JaroM_AbbTokS <=0.8 and JaroM_TokAbbSynT > 0.7 and Mon_TokSynT > 0.8 and Mon_TokS <= 0.2
- JaroM_AbbTokS <=0.8 and JaroM_TokAbbSynT > 0.7 and Mon_TokSynT <= 0.8
- JaroM_AbbTokS <=0.8 and JaroM_TokAbbSynT <= 0.7
- JaroM_AbbTokS >0.8 and Jaro_AbbTokS <= 0.9 and Jaro_TokT <= 0.9

For example, if JaroM_AbbTokS <=0.8 and JaroM_TokAbbSynT <= 0.7 then FALSE.

In Figure 4.17,

The TRUE conditions are:

- JaroM_AbbTokS > 0.9
- JaroM_AbbTokS <= 0.9 and Lev_TokSynT <= 0.7 and Jaro_TokT > 0.8 and Jaro_TokT > 0.9
- JaroM_AbbTokS <= 0.9 and Lev_TokSynT > 0.7 and Jaro_AbbTokT > 0.2
- JaroM_AbbTokS <= 0.9 and Lev_TokSynT > 0.7 and Jaro_AbbTokT <= 0.2 and Lev_TokT <= 0.1

For example, if JaroM_AbbTokS > 0.9 then TRUE.

The FALSE conditions are:

- JaroM_AbbTokS <= 0.9 and Lev_TokSynT <= 0.7 and Jaro_TokT > 0.8 and Jaro_TokT <= 0.9
- JaroM_AbbTokS <= 0.9 and Lev_TokSynT > 0.7 and Jaro_TokT <= 0.8
- JaroM_AbbTokS <= 0.9 and Lev_TokSynT > 0.7 and Jaro_AbbTokT <= 0.2 and Lev_TokT > 0.1

For example, if JaroM_AbbTokS <= 0.9 and Lev_TokSynT > 0.7 and Jaro_TokT <= 0.8 then FALSE.

Example of KB of CPR based RDR

An example of a KB of CPR based RDR created for the datasets is given in Table 4.10. In the table, the columns *RID*, *PID*, *RType*, *Condition*, *Conclusion* and *CaseID* mean rule id, parent rule id, types of rules condition for the rules, conclusion produced by rules and the classified case id respectively. Rule types GB and R represent a ground breaking rule and a refine rule respectively. A ground breaking rule is used as an alternative rule, and the conclusion of this rule is either TRUE or FALSE.

A refine rule is used as a censor rule, and the conclusion of this rule is NULL. The abbreviations *Lev*, *S*, *T*, *AbbS*, *SynT*, *TokS*, *AbbTokS*, *AbbTokT*, *TokSynT*, *Mon*, *Smith*, *Needle*, *JaroW*, *JaroM* mean Levenshtein function, source schema, target schema, abbreviation of source, synonym of target, tokenization of source, abbreviation and tokenization of source, abbreviation and tokenization of target, tokenization and synonym of target, MongeElkan, SmithWaterman, NeedlemanWunsch, JaroWinkler, JaroMeasure functions respectively. The values 1.0, 0.9, 0.7 are thresholds. For example, Lev_ST==0.8 means if the value of Levenshtein function applied to the source and target is equal to the threshold value 0.8, then the conclusion is TRUE.

Table 4.10: An example of KB for creating rules using CPR based RDR

RID	**PID**	**RType**	**Condition**	**Conclusion**	**Classified Cases**
1	0	0	0	0	0
2	1	GB	Lev_ST == 1.0	TRUE	1033
3	1	GB	Source== AbbT	TRUE	1
4	1	GB	JaroW_SynT == 1.0	TRUE	240
5	1	GB	Mon_TokS == 1.0	TRUE	204
6	1	GB	Lev_TokT== 1.0	TRUE	489
7	1	GB	Needle_ST == 0.9	FALSE	828
8	1	GB	Lev_AbbTokS >= 0.9 and JaroW_AbbTokT >= 0.9	TRUE	269
9	1	GB	Smith_TokSynT >= 0.9	TRUE	125
10	9	R	Mon_AbbTokS == 0.2 and JaroW_TokS <= 0.3	NULL	699
11	6	R	Lev_ST == 0.8	NULL	85
12	1	GB	Lev_TokSynT == 0.8	TRUE	90
13	1	GB	JaroM_ST <= 0.7	FALSE	2
14	5	R	Smith_TokSynT== 0.8	NULL	383
15	1	GB	Lev_AbbS == 0.8	TRUE	1183

In Table 4.10, rule id 1 (RID=1) is an entry rule in the KB. It is always true. For example, the rules 2 to 9 are used to classify cases of one dataset. The rules 2 to 9 are

applied to classify other datasets. In order to solve incorrect classifications of other datasets, the censor rules 10 and 11 are created by the knowledge acquisition process of the CPR based RDR approach to make the classification NULL. Following this, the alternative rules 12 and 13 are created to classify the cases as TRUE and FALSE respectively. In such a way, the rules up to 15 are created. The additions of these rules over time mean the KB is incrementally increased as new knowledge is added.

Comparison between Prune Trees and KB

I compare the prune trees of J48 (Figures 4.16 and 4.17) and KB of CPR based RDR (Table 4.10). I find that rules of the prune tree created for training one dataset are different from the rules of prune tree created for training the combinations of two datasets. If schema data changes over time, then it is necessary to recreate a training model by J48, and this model is completely different from the previous models. For the CPR based RDR approach, some rules are created to classify one dataset and the same rules are later reused to classify another new dataset. If the rules create incorrect classifications, censor and alternative rules are added for making correct classifications. So the KB is not completely different; rather the same KB with incrementally added rules are reused to classify new schema datasets. This approach incrementally increases performance in terms of precision, recall and F-measure, and it decreases rules additions.

4.5.9 Summary of Combination Approaches

I evaluated the performance of the CPR based RDR and machine learning approaches and compared the performance of the approaches. The experimental results show that CPR based RDR approach exhibits comparable performance with the machine learning approaches, and KB of CPR based RDR can be expanded incrementally as the classified cases increase. Machine learning approaches usually rebuild their models after collecting sufficient data while knowledge engineering approaches acquire new knowledge if necessary. Machine learning approach requires training datasets for building models. Obtaining suitable training datasets for large datasets is not easy.

The advantage of the CPR based RDR rule-based schema matching system compared to machine learning approaches is that it can incrementally build a KB by adding

rules to solve schema matching problems/incorrect classifications. Though it was possible to delete some false positive values and to solve the problem of false negative values using the CPR based RDR rule-based schema matching system, there are still some false positive and false negative values in the results. The limitation of the CPR based RDR approach is that it is a time-intensive task to create rules using schema features for mapping schemas one by one. For this reason, it is necessary to develop a hybrid approach that combines both machine learning and CPR based RDR approaches. In the next chapter, I describe the hybrid approach.

Chapter 5

Hybrid Approach and Experiments

In this chapter, I propose a hybrid approach called Hybrid-RDR that combines a machine learning approach with the incremental knowledge engineering approach - CPR based RDR in order to exploit the advantages and reduce the limitations of both machine learning and CPR based RDR approaches. I describe this hybrid approach in Section 5.1. In this approach, a classification model is constructed by a machine learning approach, and it is then extended by adding rules incrementally. The Hybrid-RDR works only for element-level matching. Structure-level matching is required to obtain accurate results. For this, I develop a Knowledge-based Schema Matching System (KSMS) that performs both element-level matching and structure-level matching. I describe the architecture and functionalities of KSMS in Section 5.2. I evaluate the performance of this KSMS both at the element-level and structure-level using real world datasets. I explain the evaluation processes in Section 5.3. I then show the incremental performance of the dynamic decision tree and compare this performance to KSMS at the element-level. I also compare the performance of this KSMS system to some existing systems. The last section contains a conclusion of this chapter.

5.1 Hybrid-RDR

In order to use an appropriate machine learning classification approach for the hybrid approach, I compared the performance of 11 approaches including different decision trees, rules, Naive Bayes, and AdaBootM1, described in Table 4.9. I found that pre-

cision and F-measure of Random Forest (RF) are slightly better compared to other algorithms. However, the rules obtained by RF are hard to understand directly, and it is an ensemble of unpruned classification or regression trees. Consequently, I chose J48 decision tree because it showed comparable performance to other machine learning approaches, and its model can be easily understood by a human expert. The internal nodes of the tree represent attributes, and edges represent conditions on the results of the attributes. All the leaf nodes represent classifications that are either TRUE or FALSE, indicating whether there is a match or not.

A Hybrid-RDR approach which combines the J48 decision tree with CPR based RDR is described in Figure 5.1. This hybrid approach constructs a classification model using J48 and maintains new knowledge using the CPR based RDR. When new data are available, this classification model may suggest incorrect classifications for some cases. In this case, it is necessary to add a rule called a censor rule that stops the incorrect classification of a case and to add a rule called an alternative rule that correctly classifies this incorrectly classified case. When a case is stopped by a censor rule, but there is no alternative rule that correctly classifies the case, it has no conclusion. In this case, a new rule that classifies the case correctly should be a child rule of the root node. This hybrid approach can improve the performance of schema mapping by adding rules incrementally for correcting schema matching errors of the current Knowledge Base (KB).

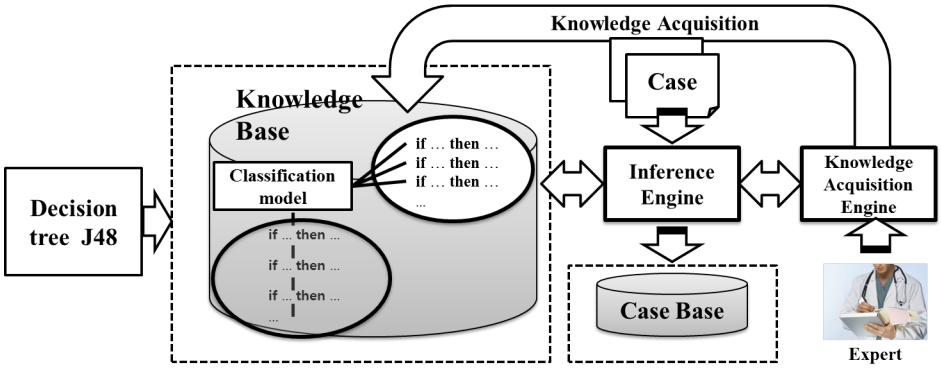

Figure 5.1: Hybrid-RDR approach

The processes of Hybrid-RDR are described below:

KB of Hybrid-RDR: The structure of the KB of Hybrid-RDR is designed as an n-ary tree. Each node of the tree is a rule, and each rule consists of IF [conditions] THEN [conclusion] UNLESS [censor-condition]. The element of [condition] of the first rule is the decision tree, J48. The elements of [condition] of other rules are object-attribute-value (OAV), and multiple [condition] elements can be combined by conjunctive [AND]. An example of the KB structure and inference process are shown in Figure 5.2.

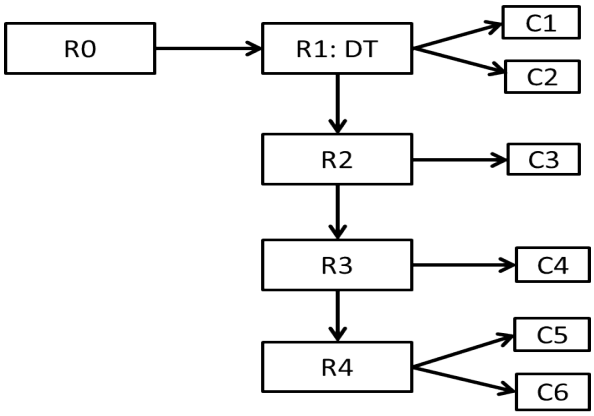

Figure 5.2: An example of KB of Hybrid-RDR

In Figure 5.2, the KB is initially empty. R0 (rule 0) is defined as a root rule. It is an entry point of the inference process, and it is always TRUE. It performs default checking such as checking the validity of the case processed. It has the NULL value for its parent link and alternative link. The first level rules are represented by R1, R2, R3 and R4. The censored/exception/stopping rules are denoted by C1, C2, C3, C4, C5 and C6. A censor rule is added if the current case conflicts with the currently fired rule. In the Hybrid-RDR approach, the first rule is added to the KB by classifying a dataset using the decision tree training model, DT. Other rules are then added incrementally for solving the schema matching/classification problems of false positives and false negatives (described in Chapter 5) produced by DT. If R1 suggests incorrect classifications, it is necessary to add censor rules C1 and C2 for making the classifications NULL. In order to correctly classify these NULL classified cases, alternative rules, for example, R2 and R3 are added to the KB as child rules of the root rule for correctly classifying the cases. If schema data changes over time, and the current KB produces incorrect classifications, censor and alternative rules will be added incrementally to the KB to classify the cases correctly.

Inference: The inference process is based on searching the KB represented as a decision list with each decision possibly refined again by another decision list (Kang et al., 1995). Once a rule is satisfied by any case, this process evaluates whether or not the censor conditions are matched to the given case. If any censor rule is not satisfied, this process stops with one path and one conclusion. However, if any censor rule is satisfied, other rules below the rule that was satisfied at the top level is evaluated. This process stops when none of the rules can be satisfied by the case in hand. This inference process for Hybrid-RDR is described below:

1. Set lastFiredRule and CurrentRule as null
2. Get censorRule of rootRule
3. If censorRule is not null, set censorRule as currentRule
4. Evaluate inputCase with currentRule
   (a) If inputCase satisfies currentRule, set currentRule as lastFiredRule and get censorRule of currentRule
      i. If censorRule is not null, set censorRule as currentRule and go to 4
   (b) Else get alternativeRule of currentRule
      i. If alternativeRule is not null, set alternativeRule as currentRule and go to 4
5. Stop inference process and return lastFiredRule

Knowledge Acquisition: Knowledge acquisition is a process that transfers knowledge from human experts to Knowledge-Based Systems (KBSs) (Kang et al., 1995). When a case is classified incorrectly, knowledge acquisition is then required. The knowledge acquisition process has been divided into three parts. Firstly, the system acquires the correct classifications from the expert. Secondly, the system decides on the new rules' locations. Thirdly, the system acquires new rules from the expert and adds them to correct the KB. If the current KB suggests an incorrect classification, it is necessary to add a censor rule for making the classification NULL. If the current KB suggests no classification for any case, a new rule should be added as an alternative rule. This rule is added as a child rule of the root node of the KB.

Cornerstone cases: The cases used to create rules are called cornerstone cases, and these cases are used in consequent knowledge acquisition process (Compton and Jansen, 1990).

Validation and Verification: The ability to easily guarantee and verify KB correctness as the expert progressively adds knowledge to the KB is a key feature of its success. This process is done by storing one cornerstone case that is used to create rules and by checking one cornerstone case per rule that ensures validity (Kang et al., 1995).

Hybrid-RDR works only for element-level matching. In order to obtain accurate results, it is necessary to do structure-level matching that considers the hierarchical structure of a full graph. For this, I developed a KSMS that performs matching both at the element-level and structure-level. The detail matching process is described in the following section.

5.2 KSMS Overview

The main components of KSMS are described in Figure 5.3. The system discovers mappings between two schemas using element-level and structure-level matchers. The final mapping results are produced by using aggregation functions. The functionalities of this system are described in Figure 5.3.

KSMS uses the following processes:

1. KSMS uses the Hybrid-RDR approach for schema mapping at the element-level. As only developing an algorithm is not sufficient to correct schema matching problems of false positives and false negatives, it is necessary to set some parameters for correcting and validating matching results. For this, I developed a GUI for the Hybrid-RDR approach that was implemented in Java. The GUI application supports selection of schemas, displaying mapping knowledge created by feature construction processes, classifying/matching elements using a J48 training model, creating rules for knowledge acquisition using features, checking satisfaction of rules, validating the schema matching results by an incremental knowledge acquisition process where rules are not predefined, and also for saving rules to the KB. A database implemented in a relational DBMS (MySQL) is used as the KB.

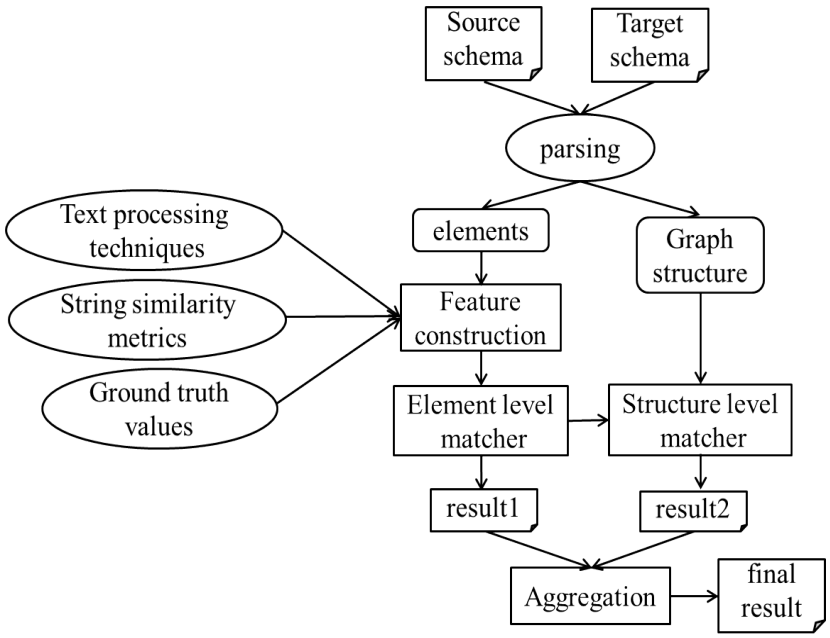

Figure 5.3: KSMS architecture

2. KSMS uses a graph matching algorithm, Similarity Flooding (Melnik et al., 2002), that matches elements considering structural information to find additional mappings.

3. The results of element-level matching and structure-level matching are combined by using aggregation functions to get final results. Finally, the performance of the aggregation functions are compared to choose the best results.

The above processes are described in detail in the following. In the KSMS system, any two schemas are first selected from a repository. At the element-level, input source and target schemas are parsed to extract names of elements. Following this, features are constructed.

5.2.1 Feature Construction

Features are constructed based on input schemas, application of string similarity metrics and string text processing techniques to the schema elements, and gold standard values. The string similarity metrics, and text processing techniques are described in Chapter 3, and feature construction processes are described in Chapter 4.

5.2.2 Element-level Matching

For element-level matching, the Hybrid-RDR approach is used. An example of the detail matching process is described below:

At the beginning of the matching process, one dataset is selected randomly for training and another dataset for testing. All the cases with 73 features (described in Chapter 4) of the test dataset are shown in the *Case Browser* in Figure 5.4.

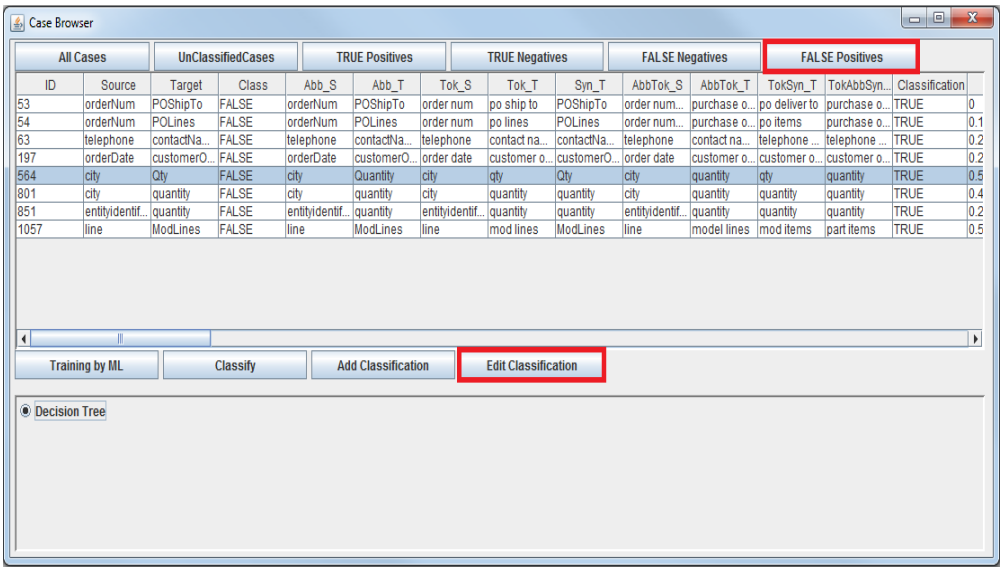

Figure 5.4: The Case Browser represents 73 features with cases (all features are not visible)

In Figure 5.4, all the cases imported from a dataset are shown in the case browser by the *All Cases* button. The *UnClassified Cases* button is used to show NULL classified cases. The Hybrid-RDR approach works in two phases: a training phase and a classification/matching phase. In the training phase, the *Training by ML* button is used to build a training model for one dataset by the decision tree J48 at the beginning. This button is used only once. The purpose of building a model is to classify whether a given element pair is matched or not based on their feature similarity measures. For J48, 10-fold cross validation is used. In the classification phase, the *Classify* button is used to classify all the cases in the *Case Browser* by the trained model. The results are found as true positive (if reported match by an expert manual mapping is TRUE and predicted match by an algorithm is TRUE), false positive (if reported

match is FALSE and predicted match is TRUE), true negative (if reported match is FALSE and predicted match is FALSE) and false negative (if reported match is TRUE and predicted match is FALSE). These are displayed by *TRUE Positives*, *FALSE Positives*, *TRUE Negatives* and *FALSE Negatives* buttons respectively.

The False positive cases are shown in Figure 5.4. The attribute *Class* provided by an expert is FALSE, but the *Classification* provided by an algorithm is TRUE. The False negative button shows that the attribute *Class* provided by an expert manually is TRUE, but the *Classification* provided by an algorithm is FALSE. The schema matching problems of false positive and false negative are solved using the knowledge acquisition process. The *Edit Classification* button is used to refine the incorrectly classified cases by adding new conditions until all incorrect cases are removed or creating another new rule using the knowledge acquisition GUI. Classification for the censor rule is always NULL. For editing the classification, the knowledge acquisition GUI is required which is displayed in Figure 5.5.

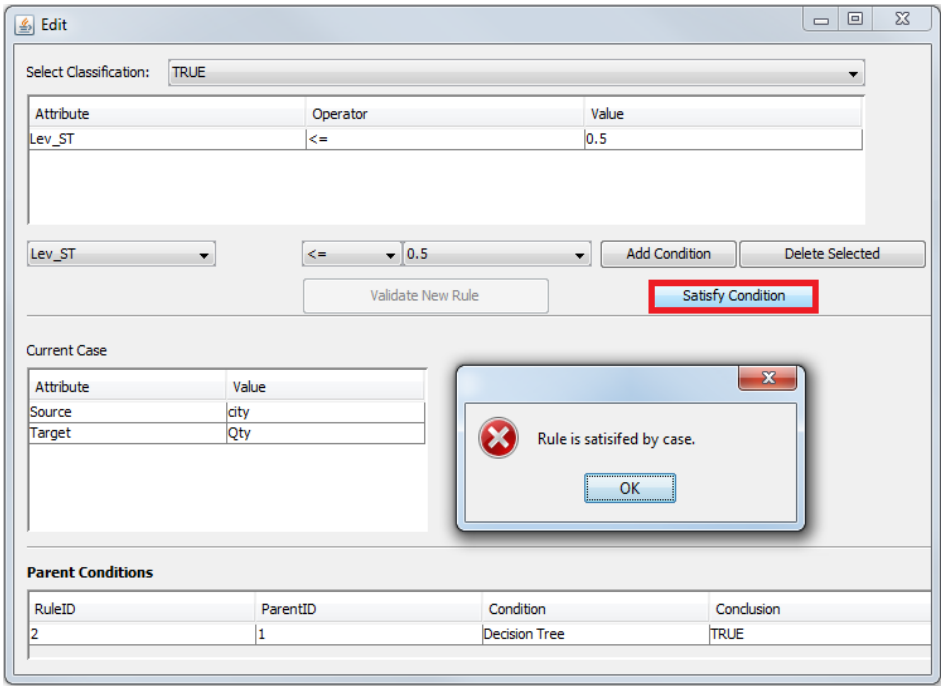

Figure 5.5: Knowledge acquisition

In Figure 5.5, the parent condition is *Decision Tree* which gives the incorrect classification for the current case. In order to edit the parent rule, it is not necessary to select the classification as the classification for the censor rules is always NULL. First,

the rule conditions are added. For each condition in the rule, *attribute*, *operator* and *value* are selected from the drop down boxes that list all the attributes, operators and values respectively. After selecting conditions, the *Add Condition* button is used to add conditions. It is possible to add more than one condition and delete a condition using the *Delete Selected* button if users think that the added condition is not appropriate. The *Satisfy Condition* button helps to look at whether the rule is satisfied by the selected case or not. In this figure, it is shown that the rule is satisfied by the current case. For this, the *Validate New Rule* button becomes active and this helps to validate the rule for all the incorrectly classified cases. The cases that satisfy the rule are shown in Figure 5.6.

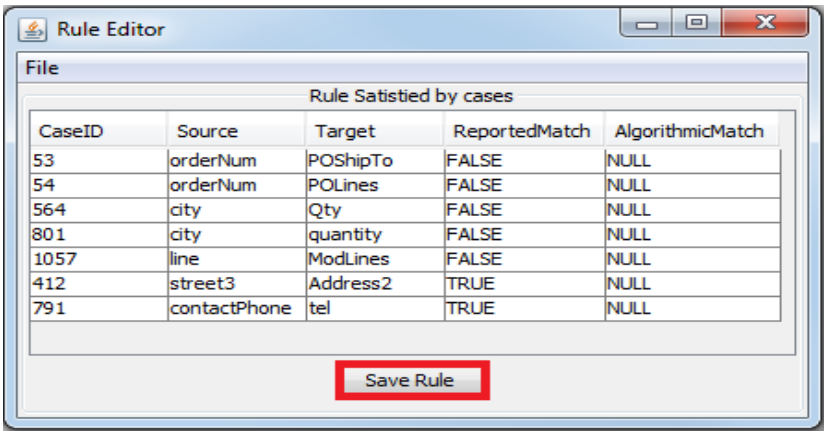

Figure 5.6: Cases that satisfy the rule

In Figure 5.6, the *Reported Match* shows the manual matching results, and the *Algorithmic Match* shows the results produced using rules. The knowledge acquisition process makes the incorrectly classified cases NULL classified. The *Save Rule* button helps to save the rule in the rule database (KB) as a censor rule and the cases in the case database. The unclassified cases are shown by the *UnClassifiedCases* button of Figure 5.4. It is seen that the attribute *Class* provided by the expert is TRUE/FALSE, but the attribute *Classification* provided by the rule is NULL.

In Figure 5.4, the *Add Classification* button is used to add alternative rules to correct the classification (TRUE or FALSE). These rules are added by the knowledge acquisition process like Figure 5.5. The difference is that there is no parent rule in this process. This is because a rule is added to classify the NULL classified cases. In order to add classification using the knowledge acquisition GUI like Figure 5.5, first

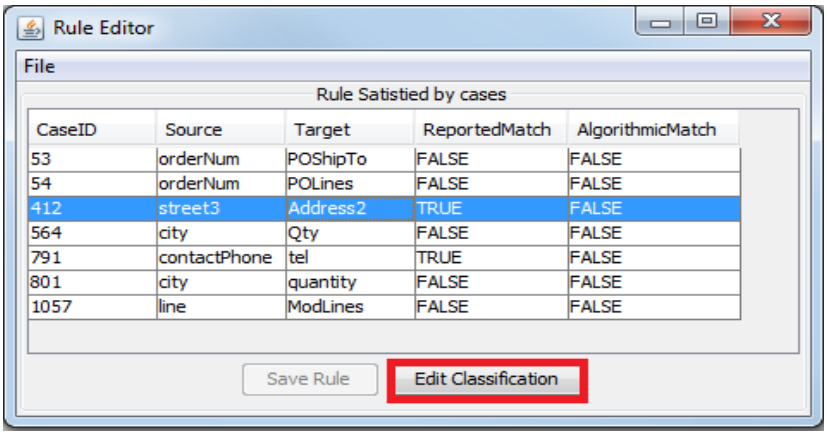

Figure 5.7: Save correct cases and edit classifications

the classification of the rule is selected. This can be done using the drop down box at the top that lists TRUE or FALSE classification for this domain. Having selected the classification, the conditions for creating a rule are added. It is then checked whether the rule is satisfied by the current case or not. If the rule is satisfied, it is then validated to determine whether the conclusion provided by the rule is matched with the reported match. As the rule is satisfied, so the *Validate New Rule* button becomes active. The cases that validate the rule are shown in Figure 5.7.

In Figure 5.7, the *Save Rule* button is used to save all the cases, the alternative rule and correct classifications. After saving rules, this *Save Rule* button becomes inactive to avoid duplicate saving. In the rule editor, some cases incorrectly satisfy the rule. So knowledge acquisition is again required to delete the cases. In this way, rules are added incrementally to the KB to solve incorrect classifications. An example of KB of the Hybrid-RDR approach to classify schema elements/cases is shown in Table 5.1

In Table 5.1, rule types GB and R represent a ground breaking rule and a refine rule respectively. A ground breaking rule is used as an alternative rule, and the conclusion of this rule is either TRUE or FALSE. A refine rule is used as a censor rule or an exception rule or a stopping rule, and the conclusion of this rule is NULL. The abbreviations *Lev*, *S*, *T*, *SynT*, *AbbSynT*, *AbbTokS*, *TokT*, *AbbT*, *TokSynT*, *AbbTokSynT*, *AbbSynS*, *TokS*, *Mon*, *Smith*, *JaroM*, *Needle* and *JaroW* mean Levenshtein function, source schema, target schema, synonym of target, abbreviation and synonym of target, abbreviation and tokenization of source, tokenization of target, abbreviation of

Table 5.1: An example of KB for classifying cases using Hybrid-RDR

RID	**PID**	**RType**	**Condition**	**Conclusion**	**Classified Cases**
1	0	0	0	0	0
2	1	GB	Decision Tree	TRUE/ FALSE	ALL
3	2	R	Lev_SynT== 1.0	NULL	964
4	1	GB	Lev_TokSynT==1.0 and JaroW_AbbTokT==0.9	TRUE	964
5	2	R	JaroW_ST == 0.9	NULL	289
6	1	GB	JaroW_ST== 0.9 and Mon_AbbSynS== 0.8	TRUE	289
7	2	R	Lev_AbbSynT == 0.8	NULL	785
8	1	GB	JaroW_TokSynT == 0.8	TRUE	785
9	2	R	JaroM_AbbTokS == 1.0 and JaroW_AbbT==0.9	NULL	1049
10	1	GB	Lev_AbbSynS == 1.0 and JaroW_AbbT==0.9	TRUE	1049
11	2	R	Lev_ST <= 0.2	NULL	567
12	1	GB	Lev_ST <= 0.2 and Smith_TokS < 0.3	FALSE	567
13	2	R	Needle_TokSynT==0.1	NULL	234
14	1	GB	Needle_AbbTokSynT==0.1	FALSE	234
15	12	R	Lev_TokT==1.0	NULL	975
16	1	GB	Lev_TokT== 1.0 and Mon_SynT== 1.0 and Smith_TokSynT >= 0.9	TRUE	975
17	6	R	Mon_TokT==0.3	NULL	640
18	1	GB	Lev_TokT==0.3	FALSE	640

target, tokenization and synonym of target, abbreviation, tokenization and synonym of target, abbreviation and synonym of source, tokenization of source, MongeElkan, SmithWaterman, JaroMeasure, NeedlemanWunsch and JaroWinkler function respectively. The values 1.0, 0.9, 0.2, 0.3, 0.8, 0.6 are thresholds. An example of a rule JaroW_ST==0.9 means if the value of JaroWinkler function applied to source and target is equal to the threshold value 0.9, the conclusion is then TRUE.

In Table 5.1, rule 1 (RID=1) is an entry rule in the KB, and it is always TRUE. The rules 2 to 18 are used to classify cases of datasets. First, rule 2 is applied to classify one dataset using a decision tree. This rule classifies the case 964 as FALSE whereas the classification provided by expert manual matching is originally TRUE. In order to solve this incorrect classification, the knowledge acquisition process is used to make the classification NULL using the rule 3. The same process is then used to create an alternatve rule 4 to classify the case as TRUE. In this way, up to 14 rules are added to the KB to solve the incorrect classifications of one dataset. Later, the rules 15 to 18 are added to the KB to solve incorrect classifications of another dataset. Adding censor rules and alternative rules incrementally build the KB.

5.2.3 Structure-level Matching

At the structure-level, input schemas are parsed and converted into a graph data structure. Structure matching is used to adjust incorrect matches generated at element-level matching, and it finds additional mappings. KSMS uses the results of element-level matching to match schema graph structures based on a well-known graph matching algorithm called Similarity Flooding (Melnik et al., 2002) that is well-known. This algorithm is described below:

Similarity Flooding: Melnik et al. (2002) presented a graph matching algorithm called Similarity Flooding (SF) and explored its usability for schema matching. This algorithm works based on the following intuition. First, schemas are converted into directed labelled graphs. These graphs are used in an iterative fixed point computation to determine the matches between corresponding nodes of graphs. Each edge in a graph contains three parts in the form $< s, p, o >$, where s and o denote the source and target nodes of the edge, and the p denotes the label of the edge. The nodes of a graph

represents schema elements (Madhavan et al., 2001). In order to compute similarities, this algorithm uses the concept that two nodes are similar when their neighbor elements are similar. This algorithm accepts several input formats, in particular, SQL DDL, XML, and RDF. The matching results produced by this algorithm is referred to as mappings.

According to Ngo et al. (2011b), many methods can be used to do structure-level matching. The methods consider that two elements of two different ontologies are similar if all or most of their elements in the same view are already similar. The view can be the direct super/sub-elements, sibling elements or list of elements in the path from the root to the current element on the ontologies' hierarchy. However, the problems of these methods are that when the viewpoints of two ontologies are similar, these methods face problems (Ngo et al., 2011b). For this, matching two nodes based on the similarity of the adjacent neighbors is more flexible and applicable.

Different kinds of neighbour elements such as parents, children and leaves, can be considered to estimate similarities between pairs of schema elements structurally. Considering only one context (parents or leaves or children) does not provide appropriate results. For this, KSMS uses the well-known Similarity Flooding algorithm considering the following three neighbouring contexts: parents, children and leaves for structure-level matching:

- **Parents:** The similarity between inner nodes of graphs is computed based on the similarity of their parent nodes. That means, two non-leaf elements are similar if they are similar according to element-level matching, and the parents of the two elements are similar.
- **Children:** The similarity of children nodes is used to determine the similarity between inner nodes of graphs. That means, two non-leaf elements are similar if they are individually similar according to element-level matching, and if the immediate children sets of the elements are similar (Madhavan et al., 2001).
- **Leaves:** The similarity of leaf nodes is used to determine the similarity between inner nodes of graphs. Two non-leaf elements are structurally similar if their leaf sets are highly similar, even if their immediate children are not (Madhavan et al., 2001). This is because the leaves represent the atomic data that the schema/ontology ultimately describes.

Graph representations of two schemas are shown in Figure 5.8 (Madhavan et al., 2001).

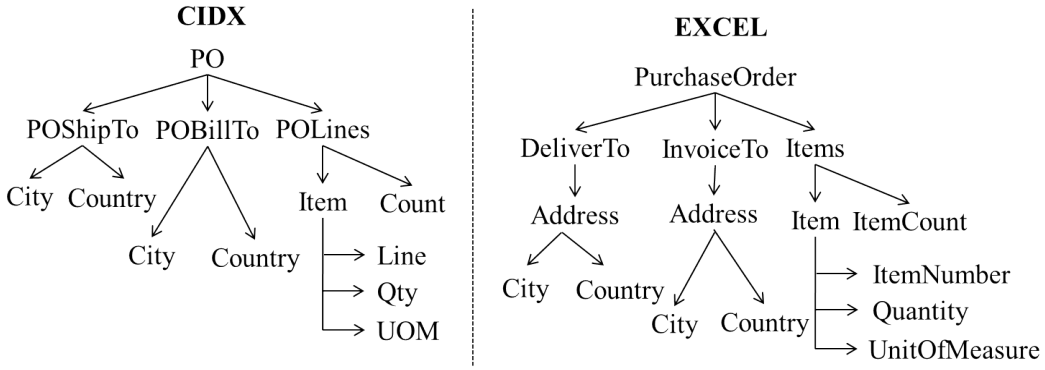

Figure 5.8: CIDX and EXCEL schemas

According to Figure 5.8, an example of structure-level matching is described below:

- *Line* is mapped to *ItemNumber* because their parents *Item* and *Item* are matched and other two children of *Item* match according to element-level matching. For example, *Qty* and *UOM* are abbreviated forms of *Quantity* and *UnitOf−Measure* respectively. So these elements are matched using domain thesaurus at the element-level.

- The children of *POShipTo* are *City* and *Country* in CIDX schema dataset. In EXCEL schema dataset, *City* and *Country* are children of *Address*, not *DeliverTo*. So *POShipTo* and *DeliverTo* can not be mapped according to children-context. However, *POShipTo* and *DeliverTo* are mapped according to the leaf-context (the leaves *City* and *Country* of *POShipTo* are matched to the leaves *City* and *Country* of *DeliverTo*) and element-level matching (*Ship* is a synonym of *Deliver*).

- The root element *PO* of CIDX is mapped to the root element of *PurchaseOrder* of EXCEL as these elements are matched according to the parent-context and element-level matching (*PO* is an abbreviated form of *PurchaseOrder*, so they are matched using domain thesaurus at the element-level).

After getting the similarities of the neighbour elements, the role of Similarity Flooding is to perform a recursive propagation of the pre-computed neighbour similarities

using fixed point computation (Hai, 2005). The Similarity Flooding algorithm terminates when the fixed point is reached, the similarity values of all schema elements stabilise and are taken as structural similarities of schema elements (Melnik et al., 2002).

5.2.4 Final Results of Mapping

In this phase, the mappings discovered from element-level matching and structure-level matching are combined by weighted, average, minimum, maximum (Djeddi and Khadir, 2014) and harmonic mean (Ngo and Bellahsene, 2012) aggregation functions to produce the final results by removing inconsistent mappings. Different systems have used different aggregation functions for combining mappings. Among them, AgreementMaker (Cruz et al., 2009) uses a linear weighted combination for obtaining unique final mapping results. Another system (Ngo et al., 2013) combines the mappings obtained by structure-level matching with mappings obtained by element-level matching to produce the set of candidate mappings by using the greedy selection method. In order to determine the most appropriate mappings, the performance of all the aggregation functions are compared, and the pairs of schema elements with maximum similarity values are chosen. The aggregation functions are described in Chapter 3.

5.3 Experimental Design

5.3.1 Datasets

The datasets used in Chapter 4 are also used in this research.

5.3.2 Experimental Procedure

In this research, I performed ten matching tasks one by one. The matching tasks D1 to D10 are described in Chapter 4. In order to use the datasets for classification and to provide appropriate knowledge to the users for knowledge acquisition, feature construction is required. The *Feature Construction* processes are described in Chapter 4.

5.3.3 Evaluation Process of the Hybrid-RDR Approach

Each dataset is processed using the Hybrid-RDR approach. The evaluation approach is shown in Figure 5.9.

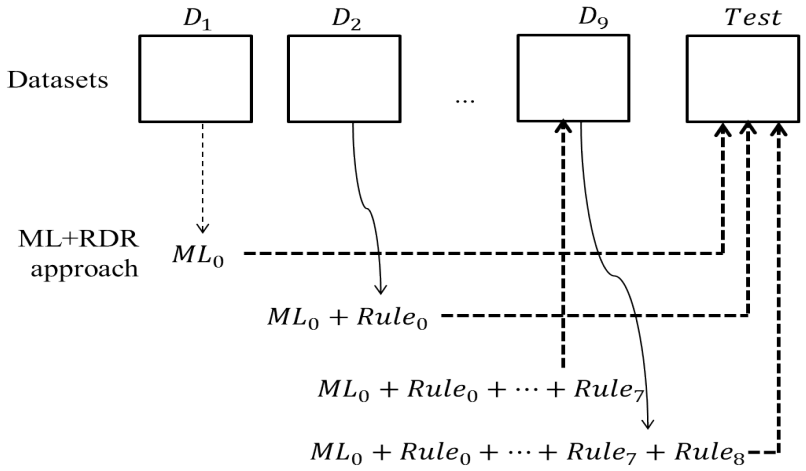

Figure 5.9: Evaluation approach of the Hybrid-RDR

In the approach, a decision tree model, ML_0, is created for one dataset and ML_0 is used to test the test dataset. Another dataset is also selected and ML_0 model is used to test this dataset. If the model finds incorrectly classified cases, then the decision tree rule is refined by censor rules, $Rule_0$, to make the incorrectly classified cases NULL classified. These NULL classified cases are again classified by alternative rules, $Rule_0$ to make the cases TRUE/FALSE classified according to gold standard. Here, the names of the censor and alternative rules are same,$Rule_0$. The conditions of this rules can be the same or different. The censor rules are added as censor nodes of the decision tree to the KB, and alternative rules are added as parent rules to the KB. The $ML_0 + Rule_0$ is reused to test the test dataset and also to test another new dataset. If any incorrectly classified case is found, then censor and alternative rules $Rule_1$ are added to correct the classification. The $ML_0 + Rule_0 + Rule_1$ is reused to test the test dataset and also to test another dataset. In this way, rules $ML_0 + Rule_0 + Rule_1 + ... + Rule_8$ are incrementally added for all nine datasets and these rules are used to test the test dataset.

Each dataset is also processed by the dynamic decision tree J48. The Evaluation approaches of the *dynamic decision tree* are shown in Section 4.5.6.

5.4 Evaluation Results

The performance of the static decision tree, dynamic decision tree and Hybrid-RDR approaches depends on the features of the datasets that are constructed using string similarity metrics and text processing techniques. The performance of the Hybrid-RDR approach also depends on the efficient knowledge acquisition process.

5.4.1 Schema Mapping Results at the Element-level

At the element-level, names of elements are matched by the dynamic decision tree J48 and Hybrid-RDR approaches. I performed ten experiments to obtain performance of the approaches. In all experiments, I randomly selected datasets for training and testing. For example, I selected $D1$ for training and $D10$ for testing, $D7$ for training and $D3$ for testing and so on. In Figure 5.10, I show how performance is increased step by step using the Hybrid-RDR and dynamic decision tree for three experiments. Precision, recall and F-measure of schema mapping using the dynamic decision tree and Hybrid-RDR, and the rules used by the Hybrid-RDR approach are shown in the figure.

In Figure 5.10, for all experiments, DDT means the results produced by the dynamic decision tree and Hybrid-RDR means the results obtained by the Hybrid-RDR approach. In addition, Exp1, Exp2 and Exp3 mean Experiment1, Experiment2, and Experiment3 respectively. In the dynamic decision tree approach, I created a decision tree model, ML_0 for one dataset and used ML_0 to test the test dataset. I then selectd another dataset and added the previous dataset for which ML_0 was created with the currently selected dataset and created ML_1 for this combined datasets. After this, I used ML_1 to test the test dataset. In this way, I created ML for all the datasets except the test dataset and used ML to test the test dataset for all ten experiments.

In the Hybrid-RDR approach, I created a decision tree model, ML_0, for one dataset and used ML_0 to test the test dataset. In this case, no rule is added. That means, rules are not pre-defined, and there is no initial expert correspondence as the KB of Hybrid-RDR is initially empty. I also selected another dataset and used ML_0 to test this dataset and obtained some incorrectly classified cases. I then refined the decision tree rule by using censor rules $Rule_0$ to make classification NULL and again classi-

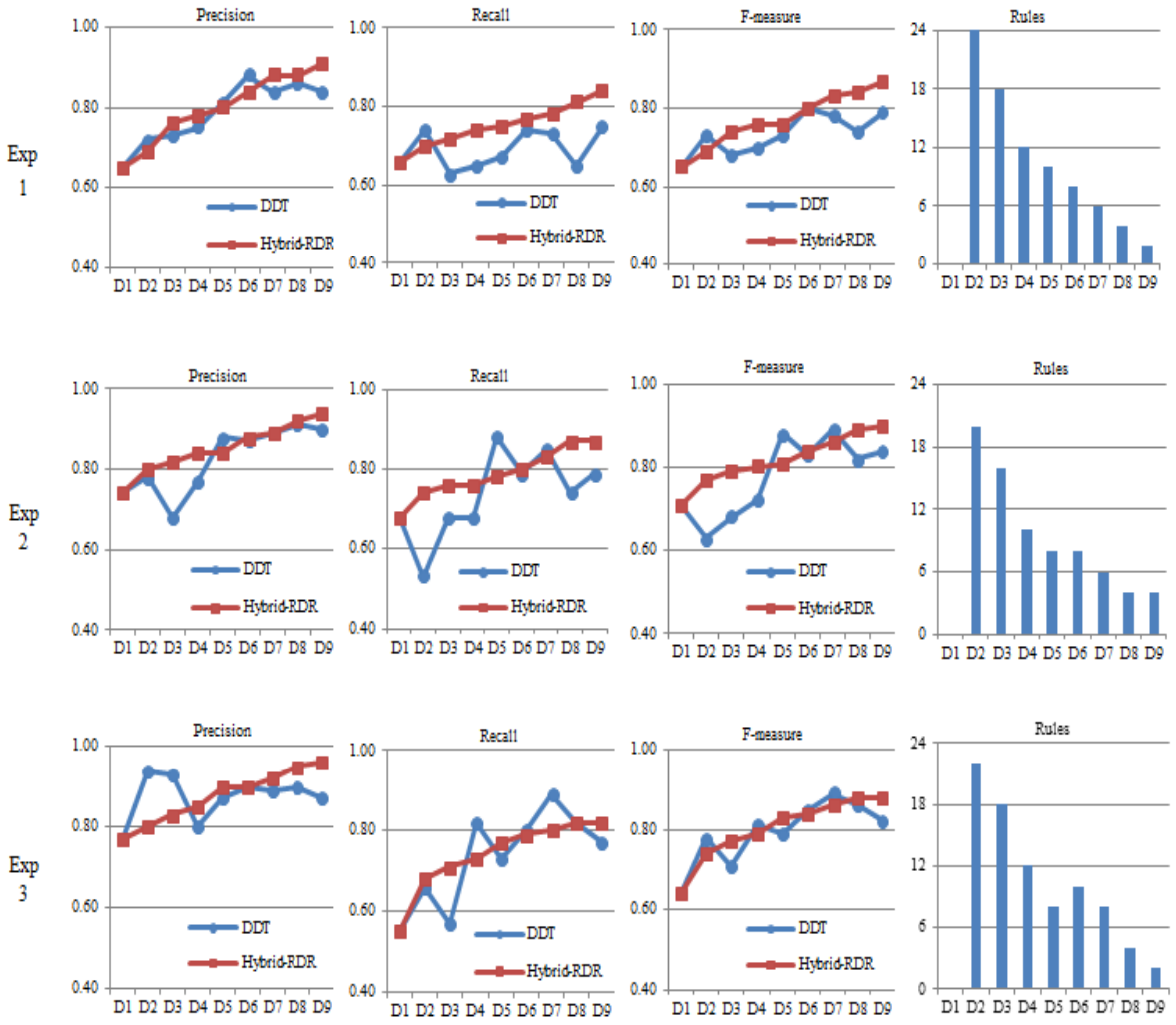

Figure 5.10: An example of incremental performance of the dynamic decision tree and Hybrid-RDR

fied the NULL classified cases by using alternative rules $Rule_0$ to make classification TRUE/FALSE. Total rules $Rule_0$ are 24, 20, 22 for experiment1, experiment2 and experiment3 respectively. The $ML_0 + Rule_0$ was reused to test the test dataset and also to test another dataset. I added rules again for the incorrectly classified cases of another dataset, and total rules $Rule_1$ are 18, 16 and 18 for experiment1, experiment2 and experiment3 respectively. The $ML_0 + Rule_0 + Rule_1$ was reused to test the test dataset and also to test another new dataset. In this way, I added rules $ML_0 + Rule_0 + Rule_1 + ... + Rule_8$ incrementally for all nine datasets and used these rules to test the test dataset. In Figure 5.10, I show that the number of rules addition

for incorrect classifications decreased gradually. This is because the decision tree model and the added rules for one dataset were reused to classify new datasets.

In order to increase the performance of DDT, it is necessary to rebuild training models with more datasets to correctly classify schema elements. However, sometimes building a model with a large number of datasets may not improve the performance by classifying schema elements correctly because of the overfitting problem of the learning approach of the decision tree (Peukert et al., 2012). Sometimes, the performance decreases significantly with the increasing sizes of decision trees. Figure 5.10 shows that although datasets were added incrementally for improving the performance of DDT, the performance sometimes increased and sometimes decreased. For example, in experiment1, 8 datasets were added to create a decision tree model, which reduced recall and F-measure by classifying schema elements for dataset D7 in this figure. However, for Hybrid-RDR, precision, recall and F-measure were improved by incrementally adding rules for adding the false negatives to and removing false positives from the final results.

In Figure 5.10, I show the incremental performance of DDT and Hybrid-RDR for only three experiments. In the same way, I performed ten experiments for each dataset and calculated the average performance of all ten experiments for each dataset. In Table 5.2, I show the average schema mapping performance of the dynamic decision tree J48 and Hybrid-RDR approaches at the element-level for all datasets. In the table, the D column describes the datasets used for the experiments, the column DDT means the results that are produced by the dynamic decision tree J48 and the column $HRDR$ means the results that are obtained by using Hybrid-RDR, and Avg means the average performance of the approaches. The results indicate that the performance is higher using the Hybrid-RDR approach compared to the performance of the dynamic decision tree J48 in almost all experiments in terms of precision, recall and F-measure. Finally, average precision, recall and F-measure for the dynamic decision tree J48 are 0.84, 0.75, and 0.78 respectively and for Hybrid-RDR are 0.89, 0.84 and 0.86 respectively.

I computed the performance of the static decision tree for matching names of elements in Chapter 4. F-measures of some approaches: AMC (Peukert et al., 2011), COMA (Do and Rahm, 2002), FALCON (Hu et al., 2008), RONDO (Melnik et al., 2003) were described by Peukert et al. (2011). These approaches are described in Chapter 3. I show the average F-measures of the AMC, COMA, FALCON, RONDO, static decision tree J48, dynamic decision tree J48 and Hybrid-RDR approaches in

Table 5.2: Average performance of the dynamic decision tree and Hybrid-RDR for element-level matching

	DDT			**HRDR**		
D	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
D1	0.86	0.85	0.85	0.94	0.85	0.90
D2	0.90	0.84	0.87	0.93	0.86	0.89
D3	0.83	0.74	0.78	0.88	0.81	0.85
D4	0.89	0.67	0.76	0.90	0.84	0.87
D5	0.89	0.77	0.82	0.94	0.80	0.86
D6	0.90	0.80	0.84	0.92	0.88	0.90
D7	0.87	0.73	0.79	0.91	0.86	0.88
D8	0.87	0.68	0.76	0.85	0.86	0.85
D9	0.71	0.81	0.75	0.80	0.88	0.83
D10	0.64	0.58	0.60	0.83	0.77	0.80
Avg	**0.84**	**0.75**	**0.78**	**0.89**	**0.84**	**0.86**

Table 5.3.

In Table 5.3, the D column describes the datasets used for the experiments, the columns, AMC, $COMA$, $FALCON$ and $RONDO$ represent F-measures of these approaches, and the column Avg means the average performance of the approaches. I denote the static decision tree, dynamic decision tree and Hybrid-RDR by SDT, DDT and HRDR respectively. The table shows a comparison of F-measures of SDT, DDT and HRDR to AMC, COMA, FALCON, and RONDO. I determine that SDT, DDT and HRDR show better performance compared to AMC, COMA, FALCON, and RONDO. The average F-measures of AMC, COMA, FALCON, and RONDO are 0.51, 0.48, 0.47 and 0.44 respectively, whereas the average F-measure for SDT is 0.68. Though SDT improves performance compared to AMC, COMA, FALCON, and RONDO, F-measure is still low.

F-measure is calculated by using precision and recall. The reason for low precision means high false positive values, and low recall means that the false negative numbers are very high (Marie and Gal, 2008). For increasing F-measure, I used the DDT that

Table 5.3: F-measures comparison of approaches

D	AMC	COMA	FALCON	RONDO	SDT	DDT	HRDR
D1	0.44	0.42	0.38	0.41	0.81	0.85	0.90
D2	0.71	0.63	0.62	0.43	0.74	0.87	0.89
D3	0.59	0.51	0.55	0.53	0.65	0.78	0.85
D4	0.52	0.46	0.35	0.47	0.62	0.76	0.87
D5	0.45	0.42	0.42	0.41	0.74	0.82	0.86
D6	0.65	0.60	0.70	0.60	0.67	0.84	0.90
D7	0.51	0.48	0.44	0.45	0.66	0.79	0.88
D8	0.55	0.50	0.54	0.55	0.64	0.76	0.85
D9	0.41	0.34	0.39	0.28	0.68	0.75	0.83
D10	0.30	0.31	0.25	0.25	0.56	0.60	0.80
Avg	**0.51**	**0.48**	**0.47**	**0.44**	**0.68**	**0.79**	**0.86**

adds datasets gradually to the previous datasets for building training models and use the models for handling false positives and false negatives. The approach improves the average F-measure up to 10% compared to the SDT, but it is still necessary to handle more false positives and false negatives to increase the performance. For this, I used HRDR that handles the problems by an efficient knowledge acquisition process. F-measure of HRDR is reasonably high compared to other approaches for all datasets. The average F-measure of HRDR is 0.86 that improves 18% and 7% compared to the SDT and DDT respectively.

The performances of all the above approaches depend on the characteristics of the datasets such as the frequency of identical, abbreviated, and synonym and combined words. If the training dataset contains a large number of abbreviated words, but the test dataset contains a large number of synonym words, the performance becomes low. For increasing the performance of the dynamic decision tree, it is necessary to build models again with more datasets to correctly classify the schema elements. Sometimes building a model with a large number of datasets using the decision tree decreases the performance because of the overfitting problems of the learning approach. However, for the Hybrid-RDR approach, performance is improved by incrementally adding rules for solving schema matching problems of false positives and false negatives.

The KSMS system uses the Hybrid-RDR approach for element-level matching. The advantages of this Hybrid-RDR approach are described below:

- In the Hybrid-RDR approach, it is not necessary to select the best combination of matchers, unlike other approaches such as COMA++ (Madhavan et al., 2005) and YAM (Duchateau et al., 2009).
- A specific threshold value is not required in the Hybrid-RDR approach, unlike other approaches such as OMReasoner (Shen et al., 2014) and AOTL (Khiat and Benaissa, 2014).
- The KB of the Hybrid-RDR approach is initially empty shown in Table 5.1. In this table, when rule id (RID) is equal to 1, the values of other columns are zero. That means, this approach does not need any initial expert correspondence from the users, unlike other approaches such as YAM (Duchateau et al., 2009).
- The first rule in the KB is decision tree J48 which classifies all the schema elements of a dataset as TRUE/FALSE. It is therefore not necessary to classify all the related schema elements one by one only by creating rules manually from the very beginning. Rules are only created to correctly classify the incorrectly classified cases produced by the J48 model. Other rules are added to the KB to add the false negatives to and to remove the false positives from the final match result. These rules are not predefined, unlike other approaches such as AMC (Peukert et al., 2011) and Self-Configuring Peukert et al. (2012). Rules are created based on the features constructed using string similarity metrics and text processing techniques.
- As only one decision tree J48 model is used to classify schema elements, this approach does not need to create training models again with more elements if schema elements change over time, and the over fitting problem does not occur, unlike another approach meta level learning (Eckert et al., 2009).
- This approach can reuse the rules. That means, the rules that are created for one dataset can be reused to match another new dataset, and new rules are added subsequently for solving the matching problems.
- The rules are not domain specific. That means, the rules that are created for purchase order domain can be reused for the same domain or another domain,

e.g., conference. For example, the rule Jaro(e1,e2) >= 0.5 satisfies the elements *customerAddress* and *Address* in the Purchase order domain, and it also satisfies other elements such as *Document* and $Conference\_document$ in the Conference domain.

- Finally, this approach supports GUI that is used to handle the schema matching problems of false positives and false negatives using the knowledge acquisition process. So users do not need to add, delete or modify schema mappings manually.

5.4.2 Schema Mapping Results at the Structure-level

In order to improve the performance by adjusting incorrect matches found from element-level matching, and produce accurate results, I performed structure-level matching. The mapping results of structure-level matching are shown in Table 5.4.

Table 5.4: Schema Mapping Results at the Structure-level

D	**Precision**	**Recall**	**F-measure**
D1	0.98	0.94	0.96
D2	0.94	0.91	0.92
D3	0.93	0.95	0.94
D4	0.97	0.94	0.95
D5	1.00	0.89	0.94
D6	0.96	0.91	0.93
D7	0.95	0.93	0.94
D8	0.91	0.94	0.92
D9	0.95	0.91	0.93
D10	0.90	0.92	0.91
Avg	**0.95**	**0.92**	**0.93**

In Table 5.4, I show the schema mapping results at structure-level matching. Precision is higher than recall in most of the datasets. This is reasonable when I consider structure-level matching instead of element-level matching. I compare this F-measure to the F-measure of element-level matching (HRDR shown in Table 5.3), and I find

that average F-measure is improved up to 7% when I consider the hierarchical structure for structure-level matching. The average precision, recall and F-measure of all the datasets of the purchase order domain are 0.95, 0.92 and 0.93 respectively.

5.4.3 Final Schema Mapping Results by Aggregation functions

To produce the final results by combining the schema mapping results produced by element-level matching and structure-level matching, I used aggregation functions. I denote the similarity values found from element-level matching and structure-level matching by *esim* and *ssim* respectively. The evaluation processes of the aggregation functions are described in Chapter 3. I have chosen the aggregation functions because most of the matching systems (Madhavan et al., 2001; Do and Rahm, 2002; Massmann and Rahm, 2008; Volz et al., 2009; Jimenez et al., 2009; Ngo and Bellahsene, 2012; Djeddi and Khadir, 2014) have used the functions. The final schema mapping results are shown in Tables 5.5, 5.6, 5.7 where the columns D, $HARM$, AVG, MIN, MAX, $WEIT$ describe information about datasets, HARMONIC MEAN, AVERAGE, MINIMUM, MAXIMUM and WEIGHTED aggregation results respectively.

Table 5.5: Final mapping results (precision)

D	HARM	AVG	MIN	MAX	WEIT
D1	0.96	0.96	0.94	0.98	0.94
D2	0.93	0.94	0.93	0.94	0.93
D3	0.90	0.90	0.88	0.93	0.88
D4	0.93	0.94	0.90	0.97	0.90
D5	0.97	0.97	0.94	1.00	0.94
D6	0.94	0.94	0.92	0.96	0.92
D7	0.93	0.93	0.91	0.95	0.91
D8	0.88	0.88	0.85	0.91	0.86
D9	0.87	0.88	0.80	0.95	0.81
D10	0.86	0.86	0.83	0.90	0.84
Avg	**0.92**	**0.92**	**0.89**	**0.95**	**0.89**

In Tables 5.5, 5.6, 5.7, I show that MAXIMUM gives the highest, and MINIMUM

gives the lowest schema mapping results compared to other aggregation functions. As MAXIMUM takes the highest value and MINIMUM takes the lowest value between two values, I do not consider the results. I compare the results among other three functions. I find that WEIGHTED provides the lowest aggregation result. The average performance of AVERAGE and HARMONIC MEAN are the same according to precision, recall and F-measure. Therefore, the final average mapping precision, recall and F-measure are 0.92, 0.88 and 0.90 respectively.

Table 5.6: Final mapping results (recall)

D	**HARM**	**AVG**	**MIN**	**MAX**	**WEIT**
D1	0.89	0.90	0.85	0.94	0.86
D2	0.88	0.88	0.86	0.91	0.86
D3	0.87	0.88	0.81	0.95	0.82
D4	0.89	0.89	0.84	0.94	0.85
D5	0.84	0.84	0.8	0.89	0.81
D6	0.89	0.90	0.88	0.91	0.88
D7	0.89	0.90	0.86	0.93	0.86
D8	0.90	0.90	0.86	0.94	0.86
D9	0.89	0.90	0.88	0.91	0.88
D10	0.84	0.84	0.77	0.92	0.78
Avg	**0.88**	**0.88**	**0.84**	**0.92**	**0.85**

I adapted KSMS for ontology mapping. In the following, I describe the evaluation process.

5.5 Ontology Mapping Evaluation

An ontology represents knowledge as a set of elements such as classes, properties and instances within a domain and the relationships between these elements (Cheatham and Hitzler, 2013a). Ontologies are designed by different people, so element heterogeneity is found in the ontologies (Ngo et al., 2011a). In order to investigate the ability of KSMS for ontology mapping, I performed evaluations on datasets of the *conference*

Table 5.7: Final mapping results (F-measure)

D	HARM	AVG	MIN	MAX	WEIT
D1	0.93	0.93	0.90	0.96	0.90
D2	0.90	0.90	0.89	0.92	0.89
D3	0.89	0.90	0.85	0.94	0.86
D4	0.91	0.91	0.87	0.95	0.87
D5	0.90	0.90	0.86	0.94	0.86
D6	0.91	0.92	0.90	0.93	0.90
D7	0.91	0.91	0.88	0.94	0.88
D8	0.88	0.88	0.85	0.92	0.86
D9	0.88	0.88	0.83	0.93	0.84
D10	0.85	0.86	0.80	0.91	0.81
Avg	**0.90**	**0.90**	**0.86**	**0.93**	**0.87**

domain.

The main difference between schema and ontology mapping is the processes of feature construction.

5.5.1 Difference between Schemas and Ontologies

The differences between the feature construction processes of schemas and ontologies are shown in Table 5.8. In the table, the *Data* row represents information of types of data: schemas and ontologies, the *Inputs* row describes the data format of the schemas and ontologies, the *Parsers* row represents different parsers used to parse schemas and ontologies, the *Element* row provides types of text used for schemas and ontologies. The text processing techniques, external resources and string similarity metrics used for constructing features of schemas and ontologies are described in the rest of the rows.

Feature Construction of Ontologies: Features of elements of ontologies are constructed using the terminological matching approaches: text processing techniques and string similarity metrics. The feature construction processes are:

Table 5.8: Differences between feature construction processes of schemas and ontologies

Processes	**Schemas**	**Ontologies**
Inputs	XML, XSD, XDR	RDF, OWL, Linked Data
Parsers	XML parser, XSD parser, XDR Parser	OWL Parser, RDF Parser
Elements	names	names, labels, comments
Text processing techniques	tokenization, abbreviation, synonym	tokenization, abbreviation, synonym, stemming, stop word removal, translation
External resources	abbreviation and synonym files created for COMA (Do and Rahm, 2002)	WordNet (Miller, 1995)
String similarity metrics	all the metrics classified by Cohen et al. (2003)	all the metrics classified by Cohen et al. (2003), Stoilos and LCS Metrics

Phase 1 Extraction of Elements of the Ontologies: In this phase, KSMS loads two ontologies from a repository and extracts their elements: names, labels, comments of the classes and properties (object properties and datatype properties) using an ontology API parser (Jena).

Phase 2 Application of Text Processing Techniques: In this phase, KSMS applies the following text processing techniques to the elements found from phase 1.

- **Tokenization:** Tokenization is described in Chapter 3.
- **Synonym matching:** In synonym matching, names of elements are matched according to their semantics using common knowledge thesaurus WordNet. For instance, elements *section* and *chapter* are matched using WordNet.
- **Stop word removal:** Stop words are used commonly in a language such as articles, prepositions or conjunction, and they add very small useful information. It is necessary to remove the words in order to increase the performance. Some examples of stop words are *above*, *a*, *am*, *is*, *are*.

- **Stemming:** Stemming is a process that is used to reduce the grammatical differences between words due to verb tense, plurals and other word forms to their base or root form. An example of stemming is *published* → *publishing*. In this case, the verb form of both words is *publish*. As both have the same verb form, the token of *published* and *publishing* are matched.

Phase 3 Application of String Similarity Metrics: In this phase, string similarity metrics are applied to the elements found from Phase 1 and 2. KSMS uses string similarity metrics developed by two open source projects. For Levenshtein, JaroWinkler, Jaro Measure, TFIDF and Jaccard, KSMS uses the open source library SecondString[1] and for MongeElkan, SmithWaterman, NeedlemanWunsch, Q-gram and Cosine, KSMS uses the SimMetric[2] open source library. In addition, for Longest Common Substring and Stoilos, KSMS uses StringMetricTester.[3] Similarity values are normalized such that the values are from 0 to 1, where 0 means strong dissimilarity and 1 means strong similarity. The threshold values for deciding the ontology elements mapping (TRUE/FALSE) are increased with 0.1 from 0 to 0.9.

Phase 4 Using Gold Standard: Another feature is created by using gold standard values (TRUE/FALSE). These features and feature values are termed *attributes* and *cases* respectively.

Applying all the above four steps, 147 features are obtained by using ontology information of each matching task. However, 73 features are obtained by using schema information of each matching task described in Chapter 4. The prepared features and feature values of each ontology matching task are used to evaluate the performance of ontology mapping.

5.5.2 Ontology Datasets

The Ontology Alignment Evaluation Initiative (OAEI)[4] was started in 2004 with the purpose of assessing strengths and weaknesses of ontology matching algorithms. Some

[1] http://secondstring.sourceforge.net
[2] http://sourceforge.net/projects/simmetrics
[3] http://www.pascal-hitzler.de/pub/StringMetricTester.jar
[4] http://oaei.ontologymatching.org/

tracks are available in different domains. In this research, I used the *conference* and *benchmark* tracks from the OAEI 2014 campaign. I selected bibliography from the benchmark tack. The conference track contains 16 ontologies. The reference alignment contains 21 alignments (test cases), which corresponds to the complete alignment space between seven ontologies (Cmt, Conference, ConfOf, Edas, Ekaw, Iasted, Sigkdd). The ontologies contain 88 (Cmt), 153 (Conference), 74 (ConfOf), 153 (Edas), 106 (Ekaw), 181 (Iasted), 177 (Sigkdd) elements. I combined two different ontologies at a time for the conference test set and obtained 21 pairs. I took the Cartesian product of elements of all the pairs. I denote the ontology pairs by D1 to D21 datasets respectively. The size of each pair is different and described in Table 5.9.

In Table 5.9, the 21 datasets are used for 21 matching tasks. I used 8 ontologies (101, 201_2, 201_4, 201_6, 201_8, 221, 237, 238) of bibliography. Among these ontologies, 101 is reference ontology and it contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals. The rest are ontologies that are systematically generated from the reference ontology. I combined 101 with other ontologies one by one and obtained 8 pairs. Finally, I took the Cartesian product of entities of all the pairs. The size of the entities of the pairs of bibliography is 10201. To use the datasets for classification and to provide appropriate knowledge to the users for knowledge acquisition, feature construction is performed. Feature construction processes of ontologies are described in Section 5.5.1. The evaluation approach used for schema mapping in Section 5.3.3 is also used for ontology mapping.

5.5.3 Ontology Mapping Results at the Element-level

Ontology mapping results for the conference track test sets are described in Table 5.10. The column *Datasets* describe the test pairs. The static decision tree, dynamic decision tree and Hybrid-RDR are represented by SDT, DDT and $HRDR$ respectively. Precision, recall and F-measure are denoted by P, R and F respectively. HARMONIC MEAN of all the results are represented by $H-mean$.

I compare precision, recall and F-measure of the SDT, DDT and HRDR. I find that precision, recall and F-measure of SDT are 0.78, 0.52 and 0.62 respectively, and these are low. In order to increase the performance, I used the DDT that adds datasets gradually for building a training model and use the model for handling false positives

Table 5.9: The sizes of ontology pairs

Ontology pairs	**Datasets**	**Elements**
Cmt-Conference	D1	10824
Cmt-ConfOf	D2	6512
Cmt-Edas	D3	13464
Cmt-Ekaw	D4	9328
Cmt-Iasted	D5	15929
Cmt-Sigkdd	D6	6776
Conference-ConfOf	D7	9102
Conference-Edas	D8	18820
Conference-Ekaw	D9	13039
Conference-Iasted	D10	22264
Conference-Sigkdd	D11	9471
ConfOf-Edas	D12	11322
ConfOf-Ekaw	D13	7845
ConfOf-Iasted	D14	13395
ConfOf-Sigkdd	D15	5698
Edas-Ekaw	D16	16218
Edas-Iasted	D17	27693
Edas-Sigkdd	D18	11781
Ekaw-Iasted	D19	19186
Ekaw-Sigkdd	D20	8162
Iasted-Sigkdd	D21	13937

Table 5.10: Ontology mapping results at the element-level (Conference)

	SDT			**DDT**			**HRDR**		
Datasets	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
D1	0.84	0.41	0.55	0.87	0.58	0.70	0.89	0.75	0.81
D2	0.90	0.45	0.60	0.88	0.69	0.77	0.92	0.74	0.82
D3	0.92	0.59	0.72	0.90	0.68	0.77	0.90	0.78	0.84
D4	0.76	0.46	0.57	0.79	0.64	0.71	0.89	0.76	0.82
D5	0.71	1.00	0.83	0.76	0.94	0.84	0.84	0.95	0.89
D6	0.93	0.56	0.70	0.89	0.74	0.81	0.94	0.82	0.88
D7	0.70	0.48	0.57	0.76	0.69	0.72	0.85	0.78	0.81
D8	0.86	0.53	0.66	0.88	0.67	0.76	0.93	0.81	0.87
D9	0.88	0.54	0.67	0.86	0.71	0.78	0.90	0.80	0.85
D10	0.69	0.70	0.69	0.67	0.92	0.78	0.85	0.89	0.87
D11	0.75	0.52	0.61	0.78	0.67	0.72	0.89	0.79	0.84
D12	0.72	0.49	0.58	0.88	0.65	0.75	0.87	0.75	0.81
D13	0.78	0.51	0.62	0.86	0.73	0.79	0.88	0.73	0.80
D14	0.78	0.47	0.59	0.81	0.66	0.73	0.85	0.77	0.81
D15	0.83	0.42	0.56	0.83	0.69	0.75	0.91	0.75	0.82
D16	0.71	0.46	0.56	0.80	0.59	0.68	0.84	0.72	0.78
D17	0.80	0.53	0.64	0.82	0.76	0.79	0.88	0.78	0.83
D18	0.71	0.49	0.58	0.90	0.64	0.75	0.86	0.76	0.81
D19	0.67	0.91	0.77	0.72	0.90	0.80	0.87	0.94	0.90
D20	0.75	0.52	0.61	0.78	0.67	0.72	0.90	0.77	0.83
D21	0.78	0.47	0.59	0.80	0.65	0.72	0.87	0.75	0.81
H-mean	**0.78**	**0.52**	**0.62**	**0.82**	**0.70**	**0.76**	**0.88**	**0.79**	**0.83**

and false negatives. The approach improves the harmonic mean of precision, recall and F-measure up to 4%, 18%, and 14% respectively compared to the SDT, but it is necessary to handle more false positives and false negatives to increase the performance. For this, I used the HRDR approach that handles the problems by the efficient knowledge acquisition process.

The performance of HRDR is reasonably high compared to other approaches for all datasets. The harmonic mean of precision, recall and F-measure of the HRDR are 0.88, 0.79, 0.83 respectively which improves F-measure 21% and 7% compared to the SDT and DDT respectively. The performance of the approaches depends on the characteristics of the ontology datasets such as the frequency of identical, abbreviated, stemmed, synonym and combined words. If a training dataset contains a large number of abbreviated words, but a test dataset contains a large number of synonym words, then the matching performance becomes low. For increasing the performance of the dynamic decision tree, it is necessary to build models again with more datasets to correctly classify the elements. Sometimes building a model with a large number of datasets decreases performance. However, for the HRDR approach, performance is improved by incrementally adding rules for overcoming the matching problems of false positive and false negative.

5.5.4 Ontology Mapping Results at the Structure-level

Only element-level matching does not produce accurate results. In order to improve the performance and produce accurate results, I performed structure-level matching. It finds additional mappings based on the existing mappings. The mapping results of structure-level matching are shown in Table 5.11.

In Table 5.11, I show the performance of structure-level matching. I compare this performance to element-level matching results obtained using the Hybrid-RDR approach. I find that the harmonic mean of precision, recall and F-measure is improved. Precision increases from 88% to 90%, recall increases from 79% to 83% and F-measure increases from 83% to 86% when I consider the hierarchical structure for structure-level matching compared to results of element-level matching strategy. It is observed in the experimental results that structural matching improves recall while maintaining a large precision.

Table 5.11: Ontology mapping results at the structure-level (Conference)

Datasets	**Precision**	**Recall**	**F-measure**
D1	0.91	0.80	0.85
D2	0.93	0.79	0.85
D3	0.92	0.83	0.87
D4	0.91	0.81	0.86
D5	0.88	0.89	0.88
D6	0.95	0.86	0.90
D7	0.87	0.82	0.84
D8	0.89	0.85	0.87
D9	0.88	0.87	0.87
D10	0.87	0.88	0.87
D11	0.90	0.83	0.86
D12	0.92	0.81	0.86
D13	0.89	0.79	0.84
D14	0.88	0.80	0.84
D15	0.93	0.78	0.85
D16	0.87	0.78	0.82
D17	0.91	0.82	0.86
D18	0.87	0.81	0.84
D19	0.90	0.96	0.93
D20	0.94	0.83	0.88
D21	0.88	0.81	0.84
H-mean	**0.90**	**0.83**	**0.86**

5.5.5 Final Ontology Mapping Results by an Aggregation Function

In order to produce the final results by combining the ontology mapping results produced by element-level matching and structure-level matching, I used an aggregation function harmonic mean on the precision, recall and F-measure. The final schema mapping results are shown in Table 5.12 where $H - mean$ means HARMONIC MEAN.

Table 5.12: Final ontology mapping results (Conference)

Datasets	**Precision**	**Recall**	**F-measure**
D1	0.90	0.77	0.83
D2	0.92	0.76	0.83
D3	0.91	0.80	0.85
D4	0.90	0.78	0.84
D5	0.86	0.92	0.88
D6	0.94	0.84	0.89
D7	0.86	0.80	0.82
D8	0.91	0.83	0.87
D9	0.89	0.83	0.86
D10	0.86	0.88	0.87
D11	0.89	0.81	0.85
D12	0.89	0.78	0.83
D13	0.88	0.76	0.82
D14	0.86	0.78	0.82
D15	0.92	0.76	0.83
D16	0.85	0.75	0.80
D17	0.89	0.80	0.84
D18	0.86	0.78	0.82
D19	0.88	0.95	0.91
D20	0.92	0.8	0.85
D21	0.87	0.78	0.82
H-mean	**0.89**	**0.81**	**0.84**

The final mapping results: precision, recall and F-measure produced by KSMS are 0.89, 0.81 and 0.84 respectively.

5.5.6 Comparison with Other Matching Systems

I compare the results of KSMS with the participants in the conference track in OAEI 2014[5] with regard to reference alignment ra1. The results are displayed in Table 5.13.

Table 5.13: Optimal results of participants in the conference track OAEI 2014

Systems	**Threshold**	**Precision**	**Recall**	**F-measure**
AML	0	0.85	0.64	0.73
LogMap	0	0.80	0.59	0.68
LogMap-C	0	0.82	0.57	0.67
XMap	0	0.87	0.49	0.63
Edna	0.81	0.76	0.49	0.60
AOT	0.66	0.80	0.47	0.59
RSDLWB	0	0.81	0.47	0.59
LogMapLite	0	0.73	0.50	0.59
OMReasoner	0	0.82	0.46	0.59
StringEquiv	0	0.80	0.43	0.56
AOTL	0.96	0.77	0.43	0.55
MassMtch	1	0.64	0.48	0.55
KSMS	**0**	**0.89**	**0.81**	**0.84**

In Table 5.13, Edna and StringEquiv are two baseline matchers that provide some contexts for understanding other matchers' performance. Among the participants, AOT, AOTL, OMReasoner, and RSDLWB perform matching only at the element-level. Other matching systems: AML, LogMap, LogMap-C, LogMapLite, MassMtch, AML, and XMap perform matching both at the element-level and structure-level. Some systems: Edna, AOT, AOTL, and MassMtch apply a threshold value to identify mappings. However, choosing a specific threshold value for all ontologies is not feasible

[5]http://oaei.ontologymatching.org/2014/conference/eval.html

as different ontologies may have different structures. Therefore, a specific threshold may not work for all ontologies. KSMS does not need any threshold value. This system considers both element-level matching and structure-level matching, and achieves higher performance in terms of precision, recall and F-measure compared to all the participants.

5.5.7 Ontology Mapping Results (bibliography)

The final mapping results of the bibliography datasets are produced by combining element-level matching and structure-level matching using harmonic mean on the precision, recall and F-measure. The final schema mapping results are shown in Table 5.14 where $H - mean$ means HARMONIC MEAN.

Table 5.14: Ontology mapping results (bibliography)

Datasets	**Precision**	**Recall**	**F-measure**
101	1.00	1.00	1.00
201_2	0.96	0.86	0.91
201_4	0.94	0.82	0.88
201_6	0.94	0.84	0.89
201_8	0.93	0.78	0.85
221	0.95	0.89	0.92
237	0.98	0.94	0.96
238	0.98	0.92	0.95
H-mean	**0.96**	**0.88**	**0.92**

The final mapping results: precision, recall and F-measure produced by KSMS using the bibliography datasets are 0.96, 0.88 and 0.92 respectively.

5.6 Summary of this Chapter

In this chapter, I described a Hybrid-RDR approach that overcomes the limitations of both decision tree J48 and CPR based RDR. I built GUI for Hybrid-RDR that per-

forms schema mapping at the element-level. However, only element-level matching is not sufficient for schema mapping. This is because it is necessary to consider the hierarchical structure of a full graph to improve the performance and to produce accurate results. For this, I developed KSMS that performs schema mapping both for element-level matching and structure-level matching. Finally, I used aggregation functions for combining the results of both element-level matching and structure-level matching. For adapting KSMS for ontology mapping, I constructed features using ontologies, string similarity metrics and text processing techniques.

I computed the performance of KSMS for schema and ontology mapping both at the element-level and structure-level. In order to show the ability of this system, I used 5 XDR datasets of the purchase order domain, 7 ontologies of the conference domain, and 8 ontologies of the bibliography domain. The experimental results show that the system determined good performance both for element-level matching and structure-level matching for both schemas and ontologies. The final schema mapping result was determined by an aggregation function.

In the Hybrid-RDR approach, it is necessary to populate the knowledge-base by creating rules by human experts to correct matches when J48 generates incorrect matches. It may not be efficient for the human experts to create rules multiple times when schema data are created over time incessantly, and thus knowledge also changes. In addition, different experts may provide different opinions about correctness or optimality of the system. Sometimes a human expert may not be available to populate the knowledge-base by providing accurate information. In order to maintain the knowledge acquisition for schema mapping without human intervention, a simulated expert is required for the evaluation of the Hybrid-RDR approach. I explain the implementation of the simulated expert for schema mapping in Chapter 6.

Chapter 6

Simulated Expert-based Hybrid Approach and Experiments

In this chapter, I propose a Simulated Expert-based Hybrid-RDR approach for schema mapping in order to reduce human involvement. This approach generates rules automatically by simulated experts to correct the incorrect matches produced by decision tree J48. I describe this hybrid approach in Section 6.1. I explain the evaluation procedure of this approach using schema datasets of the purchase order domain in Section 6.2. The Simulated Expert-based Hybrid-RDR works only for element-level matching. Structure-level matching is required to obtain accurate results. For this, I extended the KSMS to KSMS++ that performs both element-level matching and structure-level matching. I provide the evaluation results in Section 6.3. The last section contains a conclusion of this chapter.

6.1 Simulated Expert for Hybrid-RDR

A simulated expert is described by another Knowledge Based System (KBS) as a source of expertise (Compton et al., 1995) that is used to evaluate the knowledge acquisition process of the Hybrid-RDR by replacing a human expert. Knowledge is acquired from the simulated expert by using the KA method, and a new KBS is built that should have the same competence as the KBS from which the simulated expert is derived. The simulated expert whose source of expertise is a previously built expert system for

the domain is asked for the reason why a particular conclusion was reached.

In the Simulated Expert-based Hybrid-RDR approach, the C4.5 /J48 (Quinlan, 1993) machine learning based approach is used to build a KBS, which is termed a simulated expert. A classification model is created by J48 on a schema dataset, and it is then extended by adding rules incrementally. The Knowledge Base (KB) is initially empty, and the first rule is added to the KB by classifying a new schema dataset using the classification model. If the model produces incorrect matchings/classifications for some cases, rules are then added incrementally by a simulated expert using the knowledge acquisition process. The simulated expert adds censor rules to make the incorrect classification NULL and later adds alternative rules to correct the classifications, TRUE or FALSE. The censor rules are added to the KB as child rules of the parent rules. The alternative rules are added to the KB as the child rules of the root rule, and these child rules are also called parent rules. When a case is stopped by a censor rule, but there is no alternative rule that correctly classifies it, the case has no conclusion. In this case, a new rule that classifies the case correctly should be a child rule of the root node. The rules in the KB are reused later to classify another new schema dataset.

Though the simulated expert is not as creative or wise as a human expert, it can be used to assess a knowledge acquisition process at various levels of expertise. This approach significantly reduces human effort to classify the related schemas one by one by manually creating rules, and it is possible to modify the KB by adding rules without creating a model again if the J48 classification model gives incorrect classifications whenever the schema data changes over time. This approach is reuse-oriented with less human intervention. This makes the schema matching system towards automatic. The advantage of this approach is that endless repeat experiments can be done, and it is possible for the experimenters to take complete control over all the features (Compton et al., 1991).

6.1.1 Algorithm for the Simulated Expert-based Hybrid-RDR

In this section I provide an algorithm for the Simulated Expert-based Hybrid-RDR. In this algorithm, I define a classification model called modelSE, which works as a simulated expert, and another classification model called model2, that starts classifying

the remaining datasets excluding the datasets used to create model2. This Algorithm 2 is described below:

Algorithm 2: Algorithm for the Simulated Expert-based Hybrid-RDR

1. Read dataset (D) which is the combination of all datasets;
2. Build a modelSE using J48 with 20% of D;
3. Set a dataset as a test dataset;
4. Start with an empty KB;
5. Build a model2 using J48 with a dataset excluding the test dataset;
6. Accept a new dataset;
7. Match the new dataset using model2 and add the model2 as a first rule in the KB;

7.1 if model2 produces incorrect classifications for the cases of the new dataset then process the cases using the conditions of modelSE;

7.1.1 if the rule (combinations of conditions) of modelSE gives correct matching then add a censor rule of the first rule to the KB;

7.2 if model2 does not give any conclusion for matching the new dataset then process the case using the conditions of modelSE;

7.2.1 if the conditions of modelSE gives correct classification, then add an alternative rule of the root rule to the KB ;

8. Classify the test dataset using KB;
9. Go to step 6;

6.2 Experimental Design

6.2.1 Datasets

The datasets used in Chapter 4 are also used in this research.

6.2.2 Experimental Procedure

In this research, I performed experiments on ten matching tasks one by one. The matching tasks D1 to D10 are described in Section 4.5.6. In order to use the datasets

for classification and to provide appropriate knowledge to the users for knowledge acquisition, feature construction is required. The feature construction processes are described in Section 4.5.6.

6.2.3 Evaluation Process of the Simulated Expert-based Hybrid-RDR

I processed each dataset using the Simulated Expert-based Hybrid-RDR approach. The evaluation process is similar to the Hybrid-RDR approach described in Section 5.3.3. The difference between the two approaches is that rules are created by a human expert in the Hybrid-RDR approach, while rules are created by a simulated expert in the Simulated Expert-based Hybrid-RDR.

I combined schema elements of ten schema datasets and randomly selected 20% schema elements for developing KBS using the J48 algorithm. In the previous simulated expert-based applications, different experts use a distinct number of conditions that are identified for each classification (Bindoff and Kang, 2010). For example, a stupid expert uses 25% of the known conditions for making rules that may cause later errors. Moderate, clever, clever (RDR) experts use 50%, 75% and 100% of the known rule conditions respectively. However, in the schema mapping application, I determined that 20% schema elements used to make a simulated expert (known rule conditions) works appropriately to solve incorrect classifications. The KBS for a simulated expert made by the J48 algorithm is described in Figure 6.1:

The conditions in the graph in Figure 6.1 are used as expressions of expertise. These are used later for making rules to solve the schema matching problems of false positives and false negatives. The TRUE and FALSE conditions of this graph are separated so that the simulated expert can use TRUE conditions to solve false negatives and FALSE conditions to solve false positives problems. The conditions are represented in Table 6.1:

The conditions in Table 6.1 are used in the Simulated Expert-based Hybrid-RDR approach for creating censor/exception/stopping rules and alternative rules. This approach matches names of elements at the element-level. In this approach, a decision tree model, ML_0, is created by J48 for one dataset and ML_0 is used to test the test dataset. In this case, no rule is added. This means rules are not predefined, and

```
Graph digraph J48Tree {
N0 [label="Lev_TokSynT" ]
N0->N1 [label="<= 0.7"]
N1 [label="JaroW_AbbT" ]
N1->N2 [label="<= 0.8"]
N2 [label="Mon_AbbT" ]
N2->N3 [label="<= 0.7"]
N3 [label="FALSE (4099.0/3.0)" shape=box style=filled ]
N2->N4 [label="> 0.7"]
N4 [label="Needle_AbbTokS" ]
N4->N5 [label="<= 0.8"]
N5 [label="Mon_AbbTokT" ]
N5->N6 [label="<= 0.9"]
N6 [label="FALSE (18.0)" shape=box style=filled ]
N5->N7 [label="> 0.9"]
N7 [label="Mon_AbbTokS" ]
N7->N8 [label="<= 0.6"]
N8 [label="TRUE (2.0)" shape=box style=filled ]
N7->N9 [label="> 0.6"]
N9 [label="FALSE (3.0)" shape=box style=filled ]
N4->N10 [label="> 0.8"]
N10 [label="TRUE (3.0)" shape=box style=filled ]
N1->N11 [label="> 0.8"]
N11 [label="Smith_AbbT" ]
N11->N12 [label="<= 0.6"]
N12 [label="FALSE (9.0)" shape=box style=filled ]
N11->N13 [label="> 0.6"]
N13 [label="Mon_AbbTokT" ]
N13->N14 [label="<= 0.9"]
N14 [label="Smith_ST" ]
N14->N15 [label="<= 0.9"]
N15 [label="Lev_ST" ]
N15->N16 [label="<= 0.8"]
N16 [label="TRUE (2.0)" shape=box style=filled ]
N15->N17 [label="> 0.8"]
N17 [label="FALSE (8.0)" shape=box style=filled ]
N14->N18 [label="> 0.9"]
N18 [label="TRUE (6.0)" shape=box style=filled ]
N13->N19 [label="> 0.9"]
N19 [label="TRUE (30.0/1.0)" shape=box style=filled ]
N0->N20 [label="> 0.7"]
N20 [label="TRUE (52.0/3.0)" shape=box style=filled ]
}
```

Figure 6.1: KBS for a simulated expert

Table 6.1: TRUE and FALSE conditions

TRUE Conditions	**FALSE Conditions**
• Mon_AbbTokS <= 0.6	• Mon_AbbT <= 0.7
• Mon_AbbTokT > 0.9	• JaroW_AbbT <= 0.8
• Needle_AbbTokS <= 0.8	• Lev_TokSynT <= 0.7
• Mon_AbbT > 0.7	• Mon_AbbTokT <= 0.9
• JaroW_AbbT <= 0.8	• Needle_AbbTokS <= 0.8
• Lev_TokSynT <= 0.7	• Mon_AbbT > 0.7
• Needle_AbbTokS > 0.8	• Mon_AbbTokS > 0.6
• Lev_ST <= 0.8	• Mon_AbbTokT > 0.9
• Smith_ST <= 0.9	• Smith_AbbT <= 0.6
• Mon_AbbTokT <= 0.9	• JaroW_AbbT > 0.8
• Smith_ST > 0.9	• Lev_ST > 0.8
• Smith_AbbT > 0.6	• Smith_ST <= 0.9
• JaroW_AbbT > 0.8	• Smith_AbbT > 0.6
• Lev_TokSynT > 0.7	

there is no initial expert correspondence as the KB of the Simulated Expert-based Hybrid-RDR is initially empty. The approach also selects another dataset and uses ML_0 for testing and finds some incorrectly classified cases. The simulated expert then refines the decision tree rule by adding censor rules, $Rule_0$ and again classifies the cases by adding alternative rules, $Rule_0$. Total $Rule_0$ is 16 for this experiment. The $ML_0 + Rule_0$ is then reused to test the test dataset and also to test another new dataset. The simulated expert adds rules again for the incorrectly classified cases of another dataset, and total $Rule_1$ is 14. The $ML_0 + Rule_0 + Rule_1$ is reused to test the test dataset and also to test another dataset. In such a way, the simulated expert adds rules $ML_0 + Rule_0 + Rule_1 + ... + Rule_8$ incrementally for all nine datasets, and it uses the rules to test the test dataset.

6.3 Evaluation Results

The match performance of the Simulated Expert-based Hybrid-RDR approach depends on efficient knowledge acquisition. Schema mapping results using this approach are described in the following.

6.3.1 Schema Mapping Results using Simulated Expert-based Hybrid-RDR

In this research, I performed ten experiments and calculated average performance of the experiments. In all experiments, I randomly selected datasets for training and testing. I selected $D1$ for training and $D10$ for testing, $D7$ for training and $D3$ for testing and so on. For an example, a training dataset is D1, and a test dataset is D3. In Table 6.2, I show how the performance is increased step by step using the Simulated Expert-based Hybrid-RDR for one experiment. Precision, recall and F-measure of schema mapping using this approach and the rules used by this approach are described in this table. In the table, the *Datasets* column describes the datasets used for the experiments, the *Rules* column represents the average number of rules used for each experiment.

Table 6.2: Incremental performance of the Simulated Expert-based Hybrid-RDR

Datasets	**Precision**	**Recall**	**F-measure**	**Rules**
D1	0.74	0.68	0.71	0
D2+Rule	0.88	0.88	0.88	16
D4+Rule	0.89	0.94	0.93	14
D5+Rule	0.92	0.94	0.93	10
D6+Rule	0.93	0.95	0.94	2
D7+Rule	0.93	0.95	0.94	4
D8+Rule	0.97	0.97	0.97	4
D9+Rule	0.97	0.97	0.97	4
D10+Rule	0.97	0.97	0.97	2

In Table 6.2, I show the match performance of the D3 dataset. In the Simulated Expert-based Hybrid-RDR, a classification model was created using J48 for D1 and

this model was used to classify D3. Precision, recall and F-measure of D3 are 0.74, 0.68 and 0.71 respectively. This model was also used to classify D2, and it found some incorrectly classified cases. The simulated expert added 16 rules to correct these cases. Later, this model and added rules were used to classify D3. Precision, recall and F-measure of D3 were increased to 0.88, 0.88, and 0.88 respectively. This model and added rules were also used to classify D4, and they found some incorrectly classified cases. At that time, the simulated expert added 14 rules to correct the classifications. Following this, the model+16 rules+14 rules were used to classify D3. Precision, recall and F-measure were increased to 0.89, 0.97 and 0.93 respectively. In the same way, the remaining datasets were tested, and the performance of D3 was computed. In Table 6.2, I also show that the number of added rules for correcting the incorrect classifications decreased gradually. This is because the decision tree model and the added rules for one dataset were reused to classify a new dataset. In the Simulated Expert-based Hybrid-RDR approach, precision, recall and F-measure are improved by incrementally adding rules for adding false negatives to and removing false positives from the final results.

In Table 6.2, I show the incremental performance of the Simulated Expert-based Hybrid-RDR for only one experiment for dataset D3. In the same way, I performed ten experiments (randomly selecting training datasets) for the D3 dataset, and calculated the average performance of all ten experiments. For all other datasets, I performed ten experiments separately and calculated the average performance for all datasets individually. The average performance is shown in Table 6.3. In the table, the *Datasets* column describes the test sets used for the experiments, the *Rules* column represents the total number of rules used for each dataset, and *Avg* means the average performance of the approaches.

In Table 6.3, I show the average schema mapping performance of the Simulated Expert-based Hybrid-RDR approach at the element-level for all ten experiments. The results indicate that using the Simulated Expert-based Hybrid-RDR approach, the performance is very high in almost all experiments in terms of precision, recall and F-measure. Finally, precision, recall and F-measure for this approach are 0.97, 0.98 and 0.98 respectively. An example of KB of the Simulated Expert-based Hybrid-RDR approach to classify schema elements/cases is shown in Table 6.4.

In Table 6.4, rule types GB and R represent a ground breaking rule and a re-

Table 6.3: Performance of the Simulated Expert-based Hybrid-RDR for element-level matching

Datasets	**Precision**	**Recall**	**F-measure**	**Rules**
D1	0.95	0.98	0.97	60
D2	0.93	0.97	0.95	72
D3	0.97	1.00	0.99	60
D4	0.98	1.00	0.99	56
D5	1.00	0.97	0.98	60
D6	0.95	1.00	0.98	60
D7	1.00	0.98	0.99	60
D8	1.00	0.97	0.99	62
D9	0.98	1.00	0.99	60
D10	0.96	0.96	0.96	56
Avg	0.97	0.98	0.98	61

fine rule respectively. A ground breaking rule is used as an alternative rule, and the conclusion of this rule is either TRUE or FALSE. A refine rule is used as a censor rule or an exception rule or a stopping rule, and the conclusion of this rule is NULL. The abbreviations *JaroW*, *Smith*, *Lev*, *Needle*, *Mon*, *AbbT*, *ST*, *S*, *T*, *AbbTokS*, *AbbTokT* mean JaroWinkler, SmithWaterman, Levenshtein function, NeedlemanWunsch, MongeElkan, abbreviation of target, source and target schema, source schema, target schema, abbreviation and tokenization of source, abbreviation and tokenization of target respectively. The values 0.9, 0.8, 0.7, 0.6 are thresholds. An example of a rule, Lev_ST<=0.8 means if the value of Levenshtein function applied to a source and a target is less than or equal to the threshold value 0.8, then the conclusion is TRUE.

In the table, rule 1 (RID=1) is an entry rule in the KB, and it is always TRUE. In the experiment, rules 2 to 6 were used to classify cases of datasets. First, rule 2 was used to classify one dataset using a decision tree J48. This rule classified the case 476 as TRUE, whereas the classification provided by expert manual mapping was originally FALSE. In order to solve this incorrect classification, a simulated expert used the knowledge acquisition process of the Hybrid-RDR approach for making the classification NULL using rule 3. The expert then used the same process to create an

Table 6.4: An example of KB for creating rules using Simulated Expert-based Hybrid-RDR

RID	**PID**	**RType**	**Conditions**	**Conclusion**	**Classified Cases**
1	0	0	0	0	0
2	1	GB	Decision Tree	TRUE/ FALSE	ALL
3	2	R	JaroW_AbbT > 0.8 and Smith_ST <= 0.9 and Lev_ST <= 0.8 and Lev_TokSynT <= 0.7 and Needle_AbbTokS <= 0.8 and Mon_AbbTokT > 0.9 and Mon_AbbTokS <= 0.6	NULL	476
4	1	GB	Mon_AbbT <= 0.7 and Smith_ST <= 0.9 and Lev_ST > 0.8 and Lev_TokSynT <= 0.7 and Needle_AbbTokS <= 0.8 and Mon_AbbTokT > 0.9 and Mon_AbbTokS > 0.6	FALSE	476
5	2	R	Smith_ST <= 0.9 and Smith_AbbT <= 0.6 and Needle_AbbTokS <= 0.8 and Mon_AbbTokT <= 0.9 and Lev_TokSynT <= 0.7	NULL	376
6	1	GB	JaroW_AbbT > 0.8 and Lev_TokSynT > 0.7 and Needle_AbbTokS > 0.8 and Mon_AbbTokT > 0.9 and Smith_AbbT > 0.6	TRUE	376

alternative rule 4 to classify the case as FALSE. In such a way, the expert added rules to the KB to solve the incorrect classifications of one dataset.

Rules Comparison

I compare the added rules for correcting incorrect classifications by a simulated expert and a human expert in Table 6.3 and Figure 5.10 respectively. I find that the number of added rules in both the Simulated Expert-based Hybrid-RDR and Hybrid-RDR decreases gradually. However, a total number of added rules by the simulated expert are lower than the human expert. The average numbers of added rules for each dataset by the simulated expert and the human expert are 61 and 80 respectively.

Conditions per Rule Comparison

In order to determine whether a simulated expert creates more general rules than a human expert for correcting the incorrect classifications, it is necessary to look at the number of conditions per rule created by experts. From the conditions of each rule of KBs of the Simulated Expert-based Hybrid-RDR and Hybrid-RDR described in Table 6.4 and Table 5.1 respectively, I find that the number of conditions for each rule of the Simulated Expert-based Hybrid-RDR is higher than the Hybrid-RDR. The simulated expert creates on average 6 or 7 conditions per rule, whereas the human expert creates on average 2 or 3 conditions per rule. From this, I determine that the simulated expert creates more general rules than the human expert.

Time Comparison

All experiments were performed on an Intel(R) Core(TM) i5-2435M CPU @ 2.40GHZ processor with 4GB of main memory computer. A simulated expert took on average 15 seconds for knowledge acquisition for correcting incorrect classifications of one dataset. However, a human expert took on average 10 minutes for knowledge acquisition. As an example, the average time calculation of the Simulated Expert-based Hybrid-RDR for the D3 dataset is shown in Table 6.5.

Table 6.5: Average time (seconds) calculation

Datasets	**Rules**	**Time**
D1	0	0
D2+Rule	16	37
D4+Rule	14	32
D5+Rule	10	24
D6+Rule	2	5
D7+Rule	4	8
D8+Rule	4	9
D9+Rule	4	8
D10+Rule	2	4
Avg	7	15

Advantages of the Simulated Expert-based Hybrid-RDR

The Performance of the schema matching approaches depends on the quality of the results as well as execution time and usage of main memory. The Simulated Expert-based Hybrid-RDR has the same advantages as the Hybrid-RDR (see Section 5.4.1). However, it has the following additional advantages over the Hybrid-RDR approach. 1) The Hybrid-RDR increases performance solving incorrect classifications by adding rules incrementally by a human expert when the classification model created by J48 gives incorrect classifications. However, in the Simulated Expert-based Hybrid-RDR, a simulated expert generates rules automatically to correct the incorrect classifications produced by the J48 classification model. 2) Creating rules by a human expert takes a few minutes, while the simulated expert takes only a few seconds for generating rules. 3) The number of added rules by a human expert is high. In contrast, the number of added rules in the Simulated Expert-based Hybrid-RDR is lower than Hybrid-RDR because the simulated expert can create more general rules than the human expert. 4) As the number of rules is smaller compared to Hybrid-RDR, so the Simulated Expert-based Hybrid-RDR requires a lower memory space. 5) Precision, recall and F-measure of the Simulated Expert-based Hybrid-RDR approach are higher than Hybrid-RDR. Therefore, the Simulated Expert-based Hybrid-RDR increases performance in terms of

quality of results, time and memory for schema mapping. This approach is useful for large schema datasets.

6.3.2 Schema Mapping Results using KSMS++ at the Structure-level

The Simulated Expert-based Hybrid-RDR works only for element-level matching. Structure-level matching is required to adjust incorrect matches found from the element-level matching in order to find additional mappings. Therefore, I extend the KSMS (described in Chapter 5) to KSMS++. The main difference between KSMS and KSMS++ is that KSMS uses the Hybrid-RDR for schema mapping at the element-level, whereas KSMS++ uses the Simulated Expert-based Hybrid-RDR for schema mapping at the element-level. KSMS++ uses the results of the element-level matching to match schema graph structures based on a graph matching algorithm called Similarity Flooding (Melnik et al., 2002). The algorithm is described in Chapter 5. The mapping results of structure-level matching are shown in Table 6.6.

Table 6.6: Schema mapping results using KSMS++ at the structure-level

Datasets	**Precision**	**Recall**	**F-measure**
D1	0.98	0.98	0.98
D2	0.95	0.97	0.96
D3	0.98	1.00	0.99
D4	0.98	1.00	0.99
D5	1.00	0.98	0.99
D6	0.97	1.00	0.98
D7	1.00	0.98	0.99
D8	1.00	0.97	0.99
D9	0.98	1.00	0.99
D10	0.99	0.97	0.98
Avg	0.98	0.99	0.98

In Table 6.6, I show the performance of structure-level matching in terms of precision, recall and F-measure. I compare these performance metrics to the element-level

matching, and I find that average precision and recall are improved up to 1% when I use the hierarchical structure for structure-level matching. The reason for this low improvement compared to results of element-level matching is that precision and recall are very high at the element-level matching. The average precision, recall and F-measure of all the datasets of the purchase order domain are 0.98, 0.99 and 0.98 respectively.

6.3.3 Final Mapping Results

In this phase, I combined the mappings discovered from element-level matching and structure-level matching by the average aggregation function (Djeddi and Khadir, 2014) to produce the final results. The aggregation functions are described in Chapter 3. The final mapping results are shown in Table 6.7.

Table 6.7: Final schema mapping results using KSMS++

Datasets	**Precision**	**Recall**	**F-measure**
D1	0.97	0.98	0.98
D2	0.95	0.97	0.96
D3	0.98	1.00	0.99
D4	0.98	1.00	0.99
D5	1.00	0.98	0.99
D6	0.96	1.00	0.98
D7	1.00	0.98	0.99
D8	1.00	0.97	0.99
D9	0.98	1.00	0.99
D10	0.98	0.97	0.97
Avg	0.98	0.99	0.98

The final average mapping precision, recall and F-measure are 0.98, 0.99 and 0.98 respectively.

6.4 Summary of this Chapter

In order to create rules without human intervention, I proposed a Simulated Expert-based Hybrid-RDR approach. In this approach, it is not necessary to create a new model when the schema data changes over time. The model which is created by one dataset is used to match another dataset, and rules are added incrementally by a simulated expert to solve incorrect matchings of false positives and false negatives. Later, the same model and added rules are used to classify another dataset. The main facilities of the simulated expert based approach are that endless repeat experiments can be performed, and it is also possible for the experimenter to take complete control over all the variables. The approach has the same advantages as Hybrid-RDR. It has some additional advantages which are not applicable for the Hybrid-RDR. The Simulated Expert-based Hybrid-RDR does not need human experts or users for creating rules. The rule creation process by the simulated expert to solve incorrect matchings is very fast, and the results of performance metrics are high compared to Hybrid-RDR.

In this research, I also extended the previous KSMS (described in Chapter 5) to KSMS++ that performed schema mapping both at the element-level and structure-level. KSMS++ uses the Simulated Expert-based Hybrid-RDR for element-level matching and Similarity Flooding algorithm for structure-level matching. In order to show the ability of this system, I used 5 XDR datasets of the purchase order domain. The experimental results show that this system determines good performance both at the element-level and structure-level. The final schema mapping results were determined by the average aggregation function. In future, I will adapt KSMS++ for ontology mapping and perform experiments on more datasets from other domains such as conference, bibliography and anatomy.

Chapter 7

Conclusion

This chapter concludes the thesis by providing a summary of the contributions. It then presents the suggestions for future work.

7.1 Contributions

Schema matching can be performed manually, semi-automatically and automatically. Manual matching relies heavily on expert intervention, and this is extremely time intensive. If this process is completed by a user rather than an expert, results are unreliable and error prone. In addition, different experts may have different opinions about the correctness of the matching. On the other hand, automatic matching is not feasible because of the complexity of the schema elements in the datasets. Semi-automatic matching can remove the burden of manual and automatic matching processes. Semi-automatic matching can be performed at the element-level and structure-level. Element-level matching is performed by terminological matching approaches and combination approaches. I evaluated the performance of these approaches for schema mapping. The research contributions are the following:

7.1.1 Evaluation of Terminological Matching Approaches

Terminological matching approaches are string similarity metrics and text processing techniques. In this research, I thoroughly examined how string similarity metrics and

text processing techniques impact on the performance of terminological schema matching and highlighted their limitations. I first applied string similarity metrics to names of schema elements. However, the performance of these metrics was not high. In order to improve the performance, I applied to schema elements the text processing techniques with string metrics. One fixed combination did not work for matching all schema elements because these elements contain different characteristics such as the frequency of identical, combined, abbreviated and synonym words. For this, I evaluated various string similarity metrics with different thresholds and different combinations of text processing techniques. The experimental study demonstrated that the performance of terminological schema matching was significantly improved by using text processing techniques, and different combinations of text processing techniques with different string metrics provided better match results for matching names of schema elements compared to using only string metrics. However, the performance improvement was slightly different between different schema datasets because of the characteristics of the datasets, and in spite of applying all text processing techniques, some datasets exhibited low performance because of the schema matching problems of false positives and false negatives. If false positives are high, then precision becomes low, and if false negatives are high, then recall becomes low (Duchateau et al., 2009). For overcoming the matching problems, I constructed features using terminological schema matching techniques, and these features were used later by the combination approaches - knowledge engineering and machine learning.

7.1.2 Feature Construction

I constructed features using the following four steps:

- **Step 1:** I took element names of schemas and ontologies. I then performed the Cartesian products of the elements.
- **Step 2:** I applied text processing techniques to the elements found from *Step 1*.
- **Step 3:** I applied string similarity metrics to the elements found from *Step 1* and *Step 2*. I used string similarity metrics developed by three open source projects. For Levenshtein, JaroWinkler, Jaro Measure, TFIDF and Jaccard, I used open

source library SecondString[1], and for MongeElkan, SmithWaterman, Needleman-Wunsch, Q-gram and Cosine, I used SimMetric[2] open source library. In addition, for Longest Common Substring and Stoilos, I used StringMetricTester[3]. Similarity values were normalised between 0 to 1, where 0 means strong dissimilarity and 1 means strong similarity. The threshold values were increased with 0.1 from 0 to 1.

- **Step 4:** I used gold standard (TRUE/FALSE) in this step.

The constructed features were used later for checking the performance of all the combination approaches.

7.1.3 Evaluation of Machine Learning and Knowledge Engineering Approaches

For combining the terminological schema matching techniques, I evaluated the performance of 11 machine learning approaches including decision trees, rules, Naive Bayes, AdaBootM1 using the schema features and found that J48 was the most suitable approach compared to other machine learning approaches. I ran nine iterations for increasing the similarity scores between elements, and this removed some incorrect mappings. However, it took considerable time for many iterations, and each iteration required a training model with more datasets to be rebuilt when new schema data changed. Sometimes building a model with a large number of datasets does not improve the performance by classifying the schema elements correctly because of the overfitting problem of the learning approach (Peukert et al., 2012). In addition, it is very difficult to change the model by human knowledge. Further, machine learning evaluation is generally based on using data bases of cases drawn from specific domains (Cao and Compton, 2005).

In contrast, traditional knowledge engineering approaches encode human knowledge directly such that the Knowledge Base (KB) can be constructed with limited data. These approaches can acquire new knowledge if necessary. They do not require training

[1] http://secondstring.sourceforge.net
[2] http://sourceforge.net/projects/simmetrics
[3] http://www.pascal-hitzler.de/pub/StringMetricTester.jar

datasets, unlike machine learning approaches. However, they cannot use the previous matching effort to solve the matching problem of the current ones (Lee et al., 2007). They are suitable for a small and well-defined domain. For analyzing a complex domain, they generally require a time-consuming knowledge acquisition process.

In order to overcome the limitations of machine learning approaches and time-consuming knowledge acquisition, I used an incremental knowledge engineering approach, Censored Production Rules (CPR) based Ripple-Down Rules (RDR) (Kim et al., 2012) for schema mapping. This approach used the features constructed from the *Feature Construction* processes (described in Section 7.1.2) for creating rules. The experimental results showed that CPR based RDR approach showed comparable performance with the machine learning approaches. The advantage of CPR based RDR compared to machine learning approaches is that KB can be expanded incrementally as the cases classified increase. However, the limitation of the CPR based RDR approach is that it requires considerable time to create rules for mapping pairs of elements one by one. In order to overcome the limitations of the machine learning and CPR based RDR approaches, I combined both approaches for building a hybrid approach.

7.1.4 Developing a Hybrid Approach

The hybrid approach called Hybrid-RDR combines the J48 and CPR based RDR. This hybrid approach constructs a classification model using J48 and maintains new knowledge using CPR based RDR. When new data are available, the classification model suggests incorrect classifications for some cases. In this case, it is necessary to add a rule called censor/exception/stopping rule to make the classification NULL. These rules are added to the KB as parent rules. Later, alternative rules are created to provide correct classifications. In this case, a new rule that classifies the case correctly is added to the KB as a child rule of the root node.

The advantage of Hybrid-RDR approach is that only one training model is created by J48 for a small number of schema elements. The model can be used to classify a new schema dataset, and the KB is then built incrementally by adding rules to solve schema matching problems. Later, the same model and added rules can be used to classify another new schema dataset. This process helps to reduce time in two ways. Firstly, it does not need to create training models again if schema data changes over

time. Secondly, it does not start to create alternative and censor rules using schema features from the very beginning, unlike the CPR based RDR approach. It only creates the rules to add the false negatives to and to remove false positives from the final mapping results if the J48 training model produces incorrect matchings. This hybrid approach improves the performance of schema mapping by adding rules incrementally for correcting schema matching errors of the current KB. The Hybrid-RDR approach is useful where large numbers of validation cases are at hand. In this hybrid approach, the following research issues were addressed:

1. **Developing Graphical User Interface for Hybrid-RDR**

   As fully automatic schema matching is not possible, user interaction is required for the practicability and effectiveness of a matching system. For this, I built up a robust Graphical User Interface (GUI) for the Hybrid-RDR, that allows users to correct and validate schema matching results by an incremental knowledge acquisition process for mapping schema elements. To provide sufficient knowledge about a system, the system constructs schema features by the *Feature Construction* processes described in Section 7.1.2. The features are used later to classify schema elements using the Hybrid-RDR approach. Finally, the results are sent to users for correction and validation by the knowledge acquisition process. Users create exception rules to make classification NULL and alternative rules to correct classification. More than one case may satisfy one rule. So users do not need to create rules for all cases one by one. In the approach, it is not necessary for users to select the best combinations of matchers and a specific threshold value. So users do not need to gather knowledge about the matchers.

2. **Reusing Previous Match Operations**

   Reusing previous match operations has been implemented in the Hybrid-RDR approach. In this approach, the match operations are termed rules, and it is not necessary to create rules from the very beginning. The rules which are used for matching schema elements of two datasets (D1-D2) are reused later for matching elements of other new datasets (D2-D3). For example, the rule $Lev(e1, e2) >= 0.5$ satisfies the schema elements *Address* (e1) and *customerAddress* (e2), and the rule is stored in the KB. Later, this rule can be reused to satisfy other schema elements such as *ItemNumber* and *Number*. $Lev(e1, e2) >= 0.5$ means that if

the value of Levenshtein function applied to the elements e1 and e2 is greater than or equal to the threshold value 0.5, then the conclusion is TRUE. An example of previously reused match operations is shown in the KB of Hybrid-RDR in Table 5.1. Rule 1 (RID=1) is the entry rule in the KB, and it is always TRUE. I used rules 2 to 18 to classify cases of datasets. First I applied rule 2 to classify one dataset D1 using the decision tree J48. This rule classified the case 964 as FALSE whereas the classification provided by expert manual mapping was originally TRUE. So I used the knowledge acquisition process of the Hybrid-RDR approach for making the classification NULL using rule 3. Then I created an alternative rule 4 to classify the case as TRUE. In this way, I added rules up to 14 to the KB to solve the incorrect classifications of one dataset. The same rules were reused to classify another new dataset D2. When the rules produced incorrect classifications, then new censor and alternative rules were added incrementally to the KB for correcting the classifications. I added rules 15 to 18 to solve incorrect classifications of D2. All the censor and alternative rules incrementally built the KB. This Hybrid-RDR approach reduces manual effort to match large schemas. In this approach, added rules for correcting the incorrect classifications decrease gradually, but the match performance increases. The approach also reused auxiliary information such as a domain dictionary created for COMA (Do and Rahm, 2002), WordNet and gold standard vales during *Feature Construction*.

3. **Domain Dependency**

   In the Hybrid-RDR approach, rules created for one domain can be reused for other domains. This means that the rules are not domain specific. This is because rules are created by features that are constructed by string similarity metrics and text processing techniques. For example, the rule Jaro(e1,e2) >= 0.5 satisfies the elements *customerAddress* and *Address* in the Purchase order domain, and it also satisfies other elements such as *Document* and $Conference\_document$ in the Conference domain. Jaro(e1,e2) >= 0.5 means that if the value of JaroWinkler function applied to the elements e1 and e2 is greater than or equal to the threshold value 0.5, then the conclusion is TRUE.

7.1.5 Developing a Knowledge-based System

Element-level matching alone does not produce accurate results. In order to obtain accurate results, it is necessary to do structure-level matching that considers the hierarchical structure of a full graph. For this, I developed an incremental Knowledge-based Schema Matching System (KSMS) that performs schema mapping both at the element-level and structure-level. This system uses the Hybrid-RDR approach for element-level matching. For structure-level matching, this system uses Similarity Flooding (Melnik et al., 2002) algorithm that considers that two nodes are similar when their neighbor elements are similar. I considered three neighbour elements: parents, children and leaves to estimate similarities between pairs of nodes (schema elements). I also adapted KSMS for ontology mapping both at the element-level and structure-level.

In order to to show the ability of this system, I conducted experiments using 5 XDR datasets of the purchase order domain, 7 ontologies of the conference track, and 8 ontologies of the bibliography track at OAEI 2014. Experimental results showed that this system determined good performance both at the element-level and structure-level for schemas and ontologies. I compared the performance of KSMS to other systems (results of schema and ontology mapping are found from a system (Peukert et al., 2011) and participants of OAEI 2014), and found that KSMS showed better performance compared to other systems. The advantages of KSMS are the following. 1) Recreating a training model is not required when data changes over time, so the over fitting problem does not occur. 2) This system does not need to predefine rules. Rules are created based on the features constructed by using string similarity metrics and text processing techniques. These rules are not domain specific, and these are reuse-oriented. 3) This system does not need time-consuming knowledge acquisition as rules are only created to solve matching problems produced by the decision tree model. 4) This system does not need to select a specific threshold value and the best combination of matchers. Therefore, users do not need to gather proper knowledge about the domain. 5) This system allows users to correct and validate the matching results by an incremental knowledge acquisition process. Therefore, users do not need to add, delete and modify schema and ontology mapping manually. 6) This system performs matching both at the element-level and structure-level.

7.1.6 Developing a Simulated Expert-based Hybrid-RDR

In the Hybrid-RDR approach, it may not be efficient to evaluate knowledge-based systems because experts may not be available to give their time to the training of large datasets. In addition, as the schema data changes over time and thus knowledge is changed, it may not be easy for a human expert to populate the knowledge-base by providing accurate information. Further, different experts may provide different opinions about the correctness or optimality of the system. In order to overcome these difficulties, I developed a Simulated Expert-based Hybrid-RDR approach. Like the Hybrid-RDR approach, this approach combines both C4.5/J48 and CPR based RDR. The only difference between the Simulated Expert-based Hybrid-RDR and Hybrid-RDR approaches is that knowledge acquisition is performed by a simulated expert and a human expert respectively. The simulated expert is created using J48 on a small number of schema elements randomly selecting from the total number of schema elements.

In the Simulated Expert-based Hybrid-RDR approach, the KB is initially empty. A classification model is created by J48 on a schema dataset, and the first rule is added to the KB by classifying a new schema dataset using this model. If this model produces incorrect classifications, then rules are added incrementally by a simulated expert using a knowledge acquisition process of the Hybrid-RDR approach. The simulated expert adds censor rules to make the Incorrect classification NULL and later adds alternative rules to correct the classifications (TRUE/FALSE). The censor rules are added to the KB as child rules of the parent rules, and the alternative rules are added to the KB as child rules of the root rule. These alternative rules are also called parent rules. The rules in the KB are used later to classify another new schema dataset. This approach reuses the previous match operations, and it is domain independent.

The Simulated Expert-based Hybrid-RDR has the same advantages as Hybrid-RDR (see Section 5.4.1). It has some additional advantages over the Hybrid-RDR. First, rules are created automatically by a simulated expert for solving incorrect classifications where human experts or users are not required. Second, the number of conditions for each rule of the Simulated Expert-based Hybrid-RDR is higher than the Hybrid-RDR. For this, the number of added rules in the Simulated Expert-based Hybrid-RDR is lower than the Hybrid-RDR. Third, the Simulated Expert-based Hybrid-RDR is faster than Hybrid-RDR as the simulated expert takes only a few seconds for knowledge

acquisition to correct wrong classifications of one dataset whereas the human expert takes on average few minutes for knowledge acquisition. Fourth, the performance of this approach is better than Hybrid-RDR in terms of precision, recall and F-measure.

7.1.7 Extension of KSMS to KSMS++

The Simulated Expert-based Hybrid-RDR performs mapping only at the element-level. Structure-level matching is required to adjust incorrect matches found from element-level matching, and it finds additional mappings. Therefore, I extended the KSMS to KSMS++. The main difference between KSMS and KSMS++ is that KSMS uses the Hybrid-RDR for schema mapping at the element-level whereas KSMS++ uses the Simulated Expert-based Hybrid-RDR for schema mapping at the element-level. KSMS++ uses the results of element-level matching to match schema graph structures based on a graph matching algorithm called Similarity Flooding (Melnik et al., 2002). In order to show the ability of KSMS++, I used 5 XDR datasets of the purchase order domain. The experimental results showed that this system determined good performance both at the element-level and structure-level. The final schema mapping results were determined by the average aggregation function. The final schema mapping results: precision, recall and F-measure of KSMS are 0.92, 0.88 and 0.90 respectively and the results: precision, recall and F-measure of KSMS++ are 0.98, 0.99 and 0.98 respectively. Therefore, KSMS++ determined better performance compared to KSMS.

7.2 Future Work

In future, I plan to do the following:

1. Schemas often contain different information such as data types, keys, and cardinality constraints. If they are available in the schema elements, they need to be considered by a matcher to determine element similarity. Matching elements based on constraints requires a compatibility table for automatically determining the similarity between different occurrences of a constraint (Hai, 2005). The compatibility table can cover similar data types such as *string* and *varchar* or different data types such as *number* and *string*. Sometimes, constraint-based

matching generates imperfect n:m correspondences (Hai, 2005). However, data types and keys of schema datasets contain valuable information. For reducing the complexity of schema mapping, I did not consider these in this research. In future, I will evaluate the performance of my systems by considering data types and keys of schema elements. Instance-level information is also very useful for a schema element. However, a large solution space is required for instance-level matching. I therefore only consider schema-level information in this research. In future, I will consider instance-level information.

2. Schema-level mapping is required among different schema datasets to provide a unified view to users. After determining mappings between two schemas, the next step is to generate query expressions that automatically translate data instances of these schemas under an integrated schema. However, if the schema mapping information is not stored and not shared, users need to browse the large datasets up to schema-level to identify the schemas that are mapped (Jain et al., 2010b). This problem can be solved by maintaining mapping information. The preservation of mapping is called mapping provenance (Glavic et al., 2010) where provenance (Cheney et al., 2009) is defined as a term which provides information about a source or a derivation history. Mapping provenance will help to retrieve data from multiple datasets without browsing the large datasets individually. Finally, data can be used in some applications such as data integration, data exchange, data warehousing and data mining.

Appendix A

Schema and Ontology Matching Systems

I review the state of the art schema/ontology matching systems using the criteria described in Chapter 3. I consider the systems, which are related to my approach. In the following, I focus on the schema-based, instance-based, and schema and instance-based schema and ontology matching systems.

Schema-based Systems

Schema-based systems consider the meta-data available in schemas, such as names of the schema elements, comments, data types, and schema structures, to estimate similarity between pairs of schema elements. I discuss the systems chronologically from the earliest to latest date of publication:

CUPID. CUPID (Madhavan et al., 2001) is a rule-based system that combines names, data types, constraints and structural matching algorithms at the finest levels of granularity (leaf level). It considers schemas in relational and XML formats, makes context-dependent matches of a shared type definition that is used in several larger structures. In this system, some terminological matchers such as prefix and suffix, and text processing techniques such as abbreviation, synonyms, hyponyms and homonyms are used to process schemas. This system does structural matching of schema elements based on the similarity of their contexts or vicinities. It generates 1:1 and n:1 mappings between schemas. It needs user intervention for adjusting threshold weights. In this

system, a limited number of linguistic matchers by associating a thesaurus to each schema is used, but the whole power of WordNet (Miller, 1995) is not used. The whole power of WordNet is used in another schema matching system CTXMATCH (Bouquet et al., 2003).

S-Match. S-Match (Giunchiglia and Shvaiko, 2003) is a schema-based matching system which considers meanings (concepts) and positions of nodes graphs during mapping. This system takes XML schemas and ontologies as input and returns 1:1 semantic mapping between pairs of schemas. It uses 13 element-level matchers and 3 structure-level matchers. In this system, semantic matching is implemented by a decider propositional satisfiability (SAT) algorithm. This algorithm works only on Directed Acyclic Graphs (DAG's) and is-a links. SAT deciders are correct and complete decision procedures for propositional logics. SAT allows users to find all possible mappings between elements. This system uses WordNet as background knowledge. In this system, similarity relations are considered as equivalence ($=$), overlapping ($\cap$), mismatch ($\perp$), or more general/specific ($\subseteq$, $\supseteq$). One concept is equal to another if there is at least one sense of the first concept which is a synonym of the second. One concept is overlapped with the other if there are some senses in common. Two concepts are mismatched if they have no sense in common. One concept is more general than the other if and only if there exists at least one sense of the first concept that has a sense of the other as a hyponym or as a meronym. One concept is less general than the other if and only if there exists at least one sense of the first concept that has a sense of the other concept as a hypernym or as a holonym. For example, according to WordNet, the concept *ha*" is a holonym for the concept *brim* meaning *brim* is less general than *hat*. The similar functionality of CTXMATCH is applied to S-Match. However, CTXMATCH does not provide any comparative analysis of the strengths and weaknesses of its algorithm with other techniques. This system heavily depends on SAT solvers, which decrease its time efficiency (Duchateau et al., 2009).

Falcon. Falcon (Hu et al., 2008) is an automatic system that uses a divide and conquer approach for ontology matching. It handles large ontologies, which are represented in OWL and RDFS formats. This approach first partitions the ontologies with a structure-based partitioning to separate elements (classes and properties) of each ontology into a set of small clusters using the Rock agglomerative clustering algorithm (Guha et al., 2000). It then constructs blocks out of these clusters. In the second

phase, the blocks from distinct ontologies are matched based on threshold using the I-SUB string comparison technique. Finally, in the third phase, results of the so-called V-Doc (a linguistic matcher) and GMO (an iterative structural matcher) techniques are combined via sequential composition to discover alignments between the matched block pairs. According to the results of the anatomy track of OAEI-2007, the performance of Falcon was better than other systems like S-Match and COMA. For matching schemas in the contest, Falcon took several minutes to complete, but other systems took hours and even days.

Spicy. Bonifati et al. (2008) developed a Spicy system to integrate schema matching and mapping generation. This system uses schema mapping algorithms for schema matching, mapping generation and mapping verification in order to achieve good scalability and high matching quality. A structural analysis approach with electrical circuits is used to compare value correspondences among semantically related attributes. This system can work for any data models including relational, XML and OWL. It can quickly calculate the similarity among schemas and data using electrical circuits. It does not need additional input or training data for verifying the mappings. A dedicated mapping verification module is used to verify candidate mappings when lines are suggested by a schema matching module.

Meta level learning. Meta level learning (Eckert et al., 2009) was the first to recognize the need to have more schema features for creating adaptive processes. Here, the authors combine different matching systems using machine learning approaches. They use the output of different matchers and additional features about the nature of the elements to be matched as input for the learning approach. However, no suitable gold mappings are available for learning, and the learned models created by machine learning approaches often are not able to return results with a good quality. Further, the learning approach easily overfits with the learning base, and the performance decreases significantly with the increasing sizes of decision trees.

ASMOV. ASMOV (Automatic Semantic Matching of Ontologies with Verification) (Jean-Mary et al., 2009) is an automatic ontology matching approach that integrates information of bioinformatics. It handles ontologies in OWL format and outputs n:m alignment between ontology elements (classes and properties). It works in two phases: similarity calculation and semantic verification. It uses terminological matchers, struc-

tural and extensional matchers to compute similarity measures between two ontologies. For terminological matching, this approach uses tokenization, string equality and Levenshtein distance. It also uses WordNet and UMLS as background knowledge. For structure-level matching, a weighted sum of the domain and range similarities are used to compute iterative fixed point and measuring hierarchical and restriction similarities. Later, users validate the alignments produced by ASMOV. The semantic verification is completed by examining disjoint-subsumption contradiction and subsumption incompleteness. The matching process is repeated with obtained alignments input until no new correspondence is found.

Anchor-Flood. The Anchor-Flood (Seddiqui and Aono, 2009) is a dynamic partition-based matching approach. It handles large ontologies in RDFS and OWL formats and a number of concepts called anchor. It then incrementally matches neighbours such as super-concepts, sub-concepts, and siblings of each anchor until no further matches are found, thereby building small segments (fragments) out of the ontologies to be matched. This approach uses terminological matchers, structural matchers and background knowledge base (WordNet). For terminological matching, it uses tokenization, string equality and Winkler-based similarity for matching concepts. For structure-level matching, it uses internal and external similarities, and iterative anchor-based similarity propagation. The similarity between two concepts is determined by the ratio of the number of terminologically similar direct super-concepts to the number of total direct super-concepts.

Zhong et al. Approach: Zhong et al. (2009) proposed a novel Gauss Function based ontology matching approach overcoming the problems of unbalanced ontology matching where it is necessary to match an ontology describing a local domain and another ontology describing information over multiple domains. They first determine a sub-ontology (partition) from a large ontology by evaluating the sub-graph around the similar ontology where a name matcher is applied on the Cartesian product of elements to find the most similar ontology. They then perform matching between the sub-ontology and the large ontology.

AMC. AMC (Auto Mapping Core) (Peukert et al., 2011) is a schema and ontology matching framework where it is necessary for users to provide an appropriate operator from different types of operators, such as matcher, combination, selection, analyzer and

blocking operators as inputs and to investigate the individual results of each operator. Users need to gather knowledge about the operators to use this system. In order to avoid gathering knowledge, users can use a default operator. However, sometimes the operator may not handle different schemas of different domains.

LogMap. LogMap (Jimenez-Ruiz and Grau, 2011) is an ontology matching system that maps ontologies in OWL format in the biomedicine domain. This system can deal with semantically rich ontologies such as SNOMED CT, the National Cancer Institute Thesaurus (NCI), and the Foundational Model of Anatomy (FMA) containing tens of thousands of classes. It can handle highly optimized data structures for lexically and structurally indexing the input ontologies. For the lexical indexation, LogMap uses the labels of the classes in each ontology and their lexical variations for indexation, and uses WordNet or UMLS-lexicon for enriching the indexes. For the structural indexation, it uses interval labeling schema for considering hierarchical structures of the classes. The lexical indexes are used to compute an initial set of anchor mapping and to assign a confidence value to each of them. This system works like an iterative process for new mapping starting from the initial anchor mappings and using the ontologies extended class hierarchy. It detects incoherencies and uses the greedy diagnosis algorithm for repairing to improve the quality of the resulting mappings. The goal of a repairing process is to restore coherence by minimally changing the input. LogMap2 (Jimenez-Ruiz et al., 2012) is an improved version of the LogMap system that maps ontologies in the biomedicine domain. This system supports user interaction during matching (which is essential for using cases requiring very accurate mapping), scalability of large ontologies and reasoning based diagnosis. However, in both LogMap and LogMap2, reasoning-based techniques aggravate the scalability problem, which restricts their application with more effective and complex matching strategies.

The ontology modularization technique is used to detect incoherent concepts, and a repair technique is used to minimize the incoherence and removal of matches from the input alignment (Santos et al., 2015). In this system, conflict sets of mappings are computed using a depth first search in the core fragments structure. For filtering conflict sets of mappings, confidence values are used in this system. For removing conflicts sets of mapping, two approaches are applied: 1) compute all disjoint clusters of conflicting sets; and 2) compute and remove the mappings that belong to the highest number of unresolved conflicts sets using predefined depth first search. However, the

repair technique causes loss of recall, which needs to be solved.

Instance-based Matching Systems

Instance-based matching systems exploit instance information of schemas and ontologies. We describe the systems in chronological order in the following:

SEMINT. SEMINT (Semantic Integrator) (Li and Clifton, 2000) is a system that determines attribute correspondences between heterogeneous data sources using the neural network. It is an instance-based system that creates 1:1 mapping between individual attributes of two schemas represented in the relational data model. This system associates attributes, which consist of 15 constraint-based and 5 content-based criteria derived from instance values. This system uses a function to map each possible value onto the interval [0 to 1]. In this system, the Euclidean distance is used as a terminological matcher for clustering attributes of schemas. For matching, the neural network is trained on the cluster centers of one schema and is used later to match attributes of the second schema. However, the neural network does not perform well for large schemas (Clifton et al., 1998). To improve efficiency, this system identifies a match only to attribute clusters, which are selected by users. It represents a powerful and flexible approach to hybrid matching since multiple criteria can be selected and evaluated together. However, SEMINT is only instance-based, and it does not support schema-based matching. In addition, it does not support the name-based matching and graph matching for which it may be difficult to determine a useful mapping to the interval [0 to 1].

GLUE. Doan et al. (2002) developed an instances-based ontology matching system, GLUE, using machine learning approaches. This system inputs two taxonomies with their data instances in ontologies and outputs 1:1 mapping between a set of pairs of similar concepts. For element-level matching, this system uses the TFIDF for name matching and synonyms as a string pre-processing approach. This system consists of a distribution estimator, a similarity estimator and a relaxation labeler. It uses machine learning approaches like Multi-strategy learning approach as a base learner, the Naive Bayes approach to classify text, and the meta learner approach combines the prediction of the base learner and assigns weights to base learner for finding matching among a set of instances. This system works in two phases: training and matching. At the training

phase, concept classifiers are learned for the available instances of ontology. At the matching phase, the learned concept classifiers are applied to the instances of the other ontology to determine the concepts an instance is predicted to belong to. To derive concept similarities and concept correspondences, the instance-concept associations aggregated by the Jaccard-based set similarity measure are used. However, GLUE can classify small ontology sizes between 31 and 331 concepts (Rahm, 2011). In this system, users need to specify the following: the training data, a similarity measure, a learner weight and match suggestions. This system is used to match product catalogues and Web directories.

eTuner. Lee et al. (2007) developed a machine learning based approach, eTuner, to automatically tune schema matching systems to the problems. This approach handles relational schemas and considers only 1:1 mapping between schema pairs. It uses name matchers such as edit-distance and q-gram as terminological matchers. It can match a source schema against synthetic schemas, for which the ground truth mapping is known, and it can find a tuning in order to improve the matching performance of a source schema against real schemas. It needs user assistance to improve the tuning quality by getting suggestion about the domain-specific perturbation rules. As the perturbation rules are known, so the mapping between an original source schema and a perturbed schema is also known. This approach is used for semantic matches and maintaining wrappers. However, it only considers a source schema and ignores a target schema, and tunes only small to moderate size schemas. Another limitation is that the perturbation rules are static, and so the generated gold standard does not differ much for different mapping problems (Peukert et al., 2012).

Schema and Instance based Matching Systems

The schema and instance-based matching systems exploit both schema and instance information of schemas and ontologies. The systems are described below:

ARTEMIS. ARTEMIS (Castano and De Antonellis, 1999) is a rule-based approach for schema integration. It creates 1:1 mapping at the element-level and structure-level in the relational, OO and ER models. It first computes matching in the range 0 to 1 between attributes. It then completes the schema integration by clustering attributes based on the matching and later, it construct views based on the clusters. It operates

on a hybrid relational-OO model that includes names, data types, and cardinalities of attributes and target object types of attributes that refer to other objects. This tool uses name equality, synonyms, hypernyms and domain specific thesauri for matching names of schemas. In this system, domain compatibility is used as constraints-based matcher and the similarity of relationships between two entities is used as a structure-based matcher. Finally, it computes matches by a weighted sum of names and data types matching and structural matching. This is a hybrid system where multiple criteria are used to match.

Clio. Clio (Haas et al., 2005) is a semi-automatic schema matching tool that generates mappings between a source schema and a target schema in the relational and XML data models. It consists of the following: 1) a set of Schema Readers, which read a schema and translate it into an internal representation; 2) a Correspondence Engine (CE), which is used to identify matching parts of the schemas or databases; and 3) a Mapping Generator, which generates view definitions to map data in the source schema onto data in the target schema. This tool makes use of n:m element-level matching obtained from a knowledge-base or entered by a user through a Graphical User Interface (GUI). In data integration, it executes queries to transform instances of a source schema into instances of a target schema. In data exchange (or data translation), it restructures data from a source schema into a target schema. It preserves semantic relationships between a source schema and a target schema with respect to referential integrity constraints in data transformation.

mSeer. In order to improve matchability, Chai et al. (2008) proposed a rule-based schema matching system, mSeer, that inputs relational schemas and outputs 1:1 mapping between schemas. This system uses syntactic name matchers such as prefix and dictionary-based name matchers, abbreviation and acronym expansion, and synonym look up, and other categories such as numeric, data formats and categorical. For structure-level matching, the system merges two tables, splits table and merges attributes. It automatically revises a mediated schema, which can be viewed as background knowledge in the data integration applications. The authors estimate the schema matchability by adapting eTuner work (Sayyadian et al., 2005) to tune a matching system in order to maximize accuracy. For this, they solve the problem of matching future schemas to the source schema using transformation rules to the schema and data of source.

RiMOM. RiMOM (Li et al., 2009) is an instance-based and schema-based ontology matching system. It inputs ontologies in OWL format and produces 1:1 mapping as output. It is one of the first systems that implements the dynamic multi-strategy selection of matchers. Two types of similarity measure techniques are used in this system: linguistic similarity and structural similarity. At the linguistic matching, it matches names of elements consisting of labels, comments and instance values using edit-distance over element labels, vector distance among comments and instances of elements. It also uses WordNet as background knowledge. For structure-level matching, it uses the Similarity Flooding algorithm for matching concept to concept, concept to property and property to property. This system is among the best performing prototypes in the OAEI contests until 2009.

AgreementMaker. AgreementMaker (Cruz et al., 2009) is a system that comprises a number of automatic matching methods considering a conceptual and structural level of granularities, a user interface, a set of evaluation strategies, user feedback, the type of components: schema only or schema and instances. It matches ontologies of various domains such as geospatial, environmental and biomedical. It handles ontologies in XML, RDFS, OWL, N3 and outputs 1:1, 1:n, n:n and n:m mappings. Matching process consists of two phases - similarity computation and alignment selection. In the similarity computation, it uses terminological and structural matchers. The system combines matchers using three layers. In the first layer, matchers compare concept features such as labels, comments, annotations and instances, that are represented as TFIDF vectors. The features are compared using cosine, edit-distance and JaroWinkler. This system also uses WordNet as background knowledge during matching. In the second layer, structural ontology properties are matched using descendants similarity inheritance and siblings similarity contribution. Matching results of first two layers are combined and filtered based on a user-defined threshold. The final results are sent to users for feedback (approval, rejection or modification). In terms of F-measure in the oriented matching track, the best results produced by SAMBO, RiMOM and AgreementMaker in OAEI-2008, OAEI-2009 and OAEI-2010 respectively.

PRIOR+. Mao et al. (2010) proposed PRIOR+, a generic ontology matching approach that consists of three major modules: an IR-based similarity generator, an adaptive similarity filter and weighted similarity aggregator, and a neural network-based constraint satisfaction solver. This approach measures three similarities of on-

tologies in a vector space model, and estimates the harmony of each similarity upon its corresponding matrix. In order to remove the false mappings, it adaptively aggregates different similarities and uses the aggregated results to improve overall mapping performance using an interactive activation and competition (IAC) neural network-based constraint satisfaction model. This approach solves some problems of ontology mapping: 1) aggregating multiple similarities of multiple mapping strategies; 2) manually setting parameters in aggregation functions, and 3) handling ontology constraints.

GOMMA. GOMMA (Kirsten et al., 2011) is a component-based infrastructure which manages, matches and analyses many versions of different life science ontologies. It inputs ontologies in OWL, RDF formats and outputs 1:1 mapping. In this system, three functional components: Match, DIFF and Evolution use ontologies, entities and mappings. It considers both instance-level and schema-level matching. For matching ontologies, two types of matchers - metadata-based matcher (linguistic, child, path, similarity flooding algorithms) and annotation-based matcher are used. At the linguistic matching, it uses external knowledge source and different string similarity functions such as N-gram, Levenshtein and so on. For structure-level matching, it uses the Similarity Flooding (Melnik et al., 2002) algorithm. The annotation-based matchers are used to compute the similarity between two concepts of ontologies according to the degree of concepts overlap. For this, some concepts similarity functions such as Dice, Jaccard and Cosine are used. This system scaled large ontologies successfully in the OAEI 2011 competition. It supports OnEX user interface for giving facilities to the users to add and delete concepts and attributes. It is used for query ontology, mapping versions, and statistics. However, it does not support the reasoning-based diagnosis.

LILY. LILY (Wang, 2011) is an ontology matching system that inputs RDF ontology and extracts a semantic sub-graph for each element. The information of the semantic sub-graph is the basic descriptions such as identifier, label and comments; concept descriptions such as class hierarchies, related properties and instances; property description such as hierarchies (Melnik et al., 2002), domains, ranges, restrictions and related instances. It uses both linguistic and structural information in the semantic sub-graph to generate mappings. In this system, Generic Ontology Matching (GOM) for normal size ontologies and Large Scale Ontology Matching (LOM) for large scale ontologies are used to generate matching tasks. For large scale ontologies, this system uses positive and negative reduction anchors for reducing the time complexity

in matching. It also uses ontology mapping debugging for verifying and improving mapping results. The matching process proceeds in three steps: 1) extracting semantic subgraph; 2) computing alignment similarity; and 3) similarity propagation based on semantic sub-graph and a new structure-based ontology matching method (Wang and Xu, 2009). This system uses a classic image threshold selection algorithm to automatically select a threshold and generates 1:1 mapping using the stable marriage strategy. It does not use WordNet as background knowledge, and it does not consider ontology constraints. In this system, it is necessary to manually set the size of sub-graph according to different mapping tasks.

YAM++. YAM++ (Ngo and Bellahsene, 2012) is a machine learning based ontology matching system. It takes ontologies in some ontology languages such as N3, RDF and OWL as input and produces 1:1 mapping between elements as output by element-level and structure-level matchers. For element-level matching, input ontologies are parsed to extract labels and comments as elements. In this level, terminological matcher and extensional matcher are used. The terminological matcher uses different string similarity metrics for computing the similarity score between elements. It also uses machine learning approaches decision tree, Support Vector Machine and Naive Bayes for combining string similarity metrics to produce mappings at the element-level. Extensional matcher uses external instances with ontologies to find similar instances from two ontologies, and also to discover new mappings between two instances. For structure-level matching, this system uses results of the element-level matching to identify the structural similarity of elements by analyzing the positions of schemas on the hierarchy structure of ontologies. Structural similarity algorithm of Similarity Flooding (Melnik et al., 2002) is used to calculate the structural similarity. The authors also use global diagnosis optimization method (Meilicke, 2011) in terms of semantic matching to refine candidate mappings. This system has a GUI for selecting different configurations and displaying matching results.

Summary of the Schema and Ontology Matching Systems

The summary of the schema and ontology matching system is given in Table A.1. The *System* column presents names of the schema/ontology matching systems. The *Input* column represents the input format used by the systems, the *Level* column describes

whether the systems perform mappings at the schema or instance or both levels. The *Match cardinality* column provides information about the cardinality of the computed mappings produced as output by the systems. The systems which have GUI and which do not have are represented in the *GUI* column. The combination approaches used in the systems are represented in the *Approach* column. The columns *Terminological matching* and *Structural matching* describe the matching methods used on names of schema elements and hierarchical graphs respectively.

Table A.1: Summary of schema and ontology matching systems

System	**mSeer (Chai et al., 2008)**	**eTuner (Lee et al., 2007)**	**Agreement-Maker (Cruz et al., 2009)**	**ASMOV (Jean-Mary et al., 2009)**
Input	relational	relational	XML, RDFS, OWL, N3	OWL
Level	schema, instance	schema, instance	schema, instance	schema
Match cardinality	1:1	1:1	1:1, 1:n, n:n, n:m	n:m
GUI	no	no	yes	no
Approach	rule	machine learning	-	-
Terminological matching	name, abbreviation, acronym, synonym, dictionary, unit, category values	name, edit-distance, q-gram matcher	cosine, edit-distance, JaroWinkler, WordNet	tokenization, string equality, levenshtein distance, WordNet, UMLS
Structure matching	merge-two tables, split-table, merge-attributes	tune other systems	descendant and sibling similarities	interactive fixed point computation, hierarchical, restriction similarities

System	**LILY (Wang, 2011)**	**LogMap (Jimenez-Ruiz and Grau, 2011)**	**YAM++ (Ngo and Bellahsene, 2012)**	**PRIOR+ (Mao et al., 2010)**
Input	RDF	OWL	N3, RDF, OWL	XML, RDF, RDFS, OWL
Level	schema, instance	schema	schema, instance	schema, instance
Match cardinality	1:1	1:1	1:1	1:1
GUI	no	no	yes	no
Approach	-	-	decision tree, SVM, Naive Bayes	neural network
Terminological matching	linguistics	WordNet or UMLS-lexicon	string similarity metrics	edit distance, stop word removal, stemming, tokenization
Structure matching	similarity propagation based on semantic subgraphs	interval labelling schema	Similarity Flooding	propagation of original information

System	**SMB (Marie and Gal, 2008)**	**SKAT (Mitra et al., 1999)**	**TranScm (Milo and Zohar, 1998)**	**CUPID (Madhavan et al., 2001)**
Input	XSD schemas	XML, IDL, text	DTD, OO, relational, HTML, SGML	XML, relational
Level	schema	schema	schema	schema
Match cardinality	1:1	1:1, n:1	1:1	1:1, n:1
GUI	no	no	no	no
Approach	Boosting	rule	rule	-
Terminological matching	name, domain constraints	name equality, synonyms, homonyms, hypernyms	name equality, synonyms, homonyms, hypernyms	name equality, synonyms, hypernyms, homonyms, abbreviations
Structure matching	-	matching of parent nodes and children nodes	-	matching sub-trees, weighted by leaves

System	**COMA (Do and Rahm, 2002)**	**SEMINT (Li and Clifton, 2000)**	**LSD (Doan et al., 2001)**	**Embley et al. (Embley et al., 2004)**
Input	relational, XML	relational	XML	HTML tables
Level	schema, instance	instance	schema, instance	schema, instance
Match cardinality	1:1	1:1	1:1	1:1, 1:n, n:1, n:m
GUI	yes	no	no	no
Approach	-	neural-network	base learner, Naive Bayes, meta learner	decision tree
Terminological matching	string matching algorithms, abbreviation, tokenization, synonyms and hyponyms	name equality, synonyms	name, value-characteristics and data-frame matchers	edit-distance, stop word removal, stemming, tokenization
Structure matching	children and leaf matching	-	XML classifier for matching non-leaf elements	domain ontology snippets

System	**Match-Planner (Duchateau et al., 2008)**	**Automatch (Berlin and Motro, 2002)**	**iMAP (Dhamankar et al., 2004)**	**ARTEMIS (Castano and De Antonellis, 1999)**
Input	XML schemas	relational	relational	relational, OO and ER modes
Level	schema	schema	schema	schema, instance
Match cardinality	1:1	1:1	1:1	1:1
GUI	no	no	no	no
Approach	decision tree	Naive Bayesian	machine learning, statistics and heuristics methods	rule
Terminological matching	name matching by string similarity metrics, WordNet	name, dictionary	name, domain knowledge	name equality, synonyms, hypernyms and domain specific thesauri
Structure matching	-	-	matching two nodes if one node is the successor of the other	relationships of entities

System	**GOMMA (Kirsten et al., 2011)**	**Malform-SVN (Ichise, 2008)**	**Anchor-Flood (Seddiqui and Aono, 2009)**	**RiMOM (Li et al., 2009)**
Input	OBO, OWL, RDF	OWL	RDFS, OWL	OWL
Level	schema, instance	schema, instance	schema	schema, instance
Match cardinality	1:1	1:1	1:1	1:1
GUI	yes	no	yes	no
Approach	rule	SVM	-	-
Terminological matching	N-gram, Levenshtein,external knowledge sources	string similarity metrics and word net	tokenization, string equality and Winkler-based similarity, WordNet	edit distance, vector distance, WordNet
Structure matching	Similarity Flooding	path list and word list	internal, external similarities; iterative anchor-based similarity	Similarity Flooding

References

Al-Ghanim, M., Noah, S. A., and Sembok, T. (2011). Automating XML Schema Matching: A Composite Approach. In *International Conference on Electrical Engineering and Informatics (ICEEI)*, pages 1–6. IEEE.

Alpaydin, E. (2010). *Introduction to Machine Learning (Second Edition)*, chapter 1, pages 3–14. The MIT Press, Cambridge, Massachusetts, London, England.

Amano, S., Libkin, L., and Murlak, F. (2009). XML Schema Mappings. In *Proceedings of the 28th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 33–42. ACM.

Arenas, M. and Libkin, L. (2005). XML Data Exchange: Consistency and Query Answering. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 13–24.

Athanassoulis, M. and Ailamaki, A. (2014). BF-Tree: Approximate Tree Indexing. In *Proceedings of the VLDB Endowment*, volume 7, pages 1881–1892.

Aumueller, D., Do, H.-H., Massmann, S., and Rahm, E. (2005). Schema and Ontology Matching with COMA++. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 906–908. ACM.

Barbosa, D., Freire, J., and Mendelzon, A. O. (2004). Information Preservation in XML-to-Relational Mappings. In *Database and XML Technologies*, pages 66–81. Springer.

Bellahsene, Z., Bonifati, A., and Rahm, E. (2011). *Schema Matching and Mapping*, chapter 1, page 3. Springer.

Berlin, J. and Motro, A. (2002). Database Schema Matching Using Machine Learning with Feature Selection. In *Advanced Information Systems Engineering*, pages 452–466. Springer.

Bernstein, P. A., Madhavan, J., and Rahm, E. (2011). Generic Schema Matching, Ten Years Later. In *Proceedings of the VLDB Endowment*, volume 4, pages 695–701.

Beydoun, G. and Hoffmann, A. (1997). NRDR for the Acquisition of Search Knowledge. In *Advanced Topics in Artificial Intelligence*, pages 177–186. Springer.

Beyer, K., Ozcan, F., Saiprasad, S., and Van der Linden, B. (2005). DB2/XML: Designing for Evolution. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 948–952. ACM.

Bindoff, I. and Kang, B. H. (2010). Simulated Assessment of Ripple Round Rules. In *Knowledge Management and Acquisition for Smart Systems and Services*, pages 180–194. Springer.

Bindoff, I. K. (2010). *Multiple Classification Ripple Round Rules: Classifications as Conditions*. PhD thesis, University of Tasmania.

Bohannon, P., Fan, W., Flaster, M., and Narayan, P. (2005). Information Preserving XML Schema Embedding. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 85–96. VLDB Endowment.

Bonifati, A., Mecca, G., Pappalardo, A., Raunich, S., and Summa, G. (2008). Schema Mapping Verification: The Spicy Way. In *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, pages 85–96. ACM.

Bouquet, P., Magnini, B., Serafini, L., and Zanobini, S. (2003). A SAT-Based Algorithm for Context Matching. In *Modeling and Using Context*, pages 66–79. Springer.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

Buchanan and Shortliffe (1984). *Rule-Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Buchanan, B. G. and Duda, R. O. (1983). Principles of rule-based expert systems. *Advances in Computers*, 22:163–216.

Cao, T. M. and Compton, P. (2005). A simulation Framework for Knowledge Acquisition Evaluation. In *Proceedings of the 28th Australasian Conference on Computer Science*, volume 38, pages 353–360. Australian Computer Society, Inc.

Castano, S. and De Antonellis, V. (1999). A schema Analysis and Reconciliation Tool Environment for Heterogeneous Databases. In *Proceedings of the International Database Engineering and Applications Symposium (IDEAS),*, pages 53–62. IEEE.

Cate, B. T., Dalmau, V., and Kolaitis, P. G. (2013). Learning Schema Mappings. *ACM Transactions on Database Systems (TODS)*, 38(4):28.

Chai, X., Sayyadian, M., Doan, A., Rosenthal, A., and Seligman, L. (2008). Analyzing and Revising Data Integration Schemas to Improve Their Matchability. In *Proceedings of the VLDB Endowment*, volume 1, pages 773–784.

Cheatham, M. and Hitzler, P. (2013a). String Similarity Metrics for Ontology Alignment. In *The Semantic Web - ISWC 2013*, pages 294–309. Springer.

Cheatham, M. and Hitzler, P. (2013b). The Role of String Similarity Metrics in Ontology Alignment. *Computer Science and Engineering Faculty, Wright State University*, pages 1–54.

Cheney, J., Chiticariu, L., and Tan, W.-C. (2009). Provenance in Databases: Why, How, and Where. *Foundations and Trends in Databases*, 1(2):379–474.

Cheng, W., Lin, H., and Sun, Y. (2005). An Efficient Schema Matching Algorithm. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 972–978. Springer.

Clifton, C., Housman, E., and Rosenthal, A. (1998). Experience with a Combined Approach to Attribute-Matching Across Heterogeneous Databases. In *Data Mining and Reverse Engineering*, pages 428–451. Springer.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.

Cohen, W. W., Ravikumar, P. D., and Fienberg, S. E. (2003). A Comparison of String Distance Metrics for Name-Matching Tasks. In *International Workshop on Information Integration on the Web (IIWeb)*, volume 2003, pages 73–78.

Compton, P. and Cao, T. M. (2006). Evaluation of Incremental Knowledge Acquisition with Simulated Experts. In *AI 2006: Advances in Artificial Intelligence*, pages 39–48. Springer.

Compton, P., Edwards, G., Kang, B., Lazarus, L., Malor, R., Menzies, T., Preston, P., Srinivasan, A., and Sammut, C. (1991). Ripple Down Rules: Possibilities and Limitations. In *Proceedings of the Sixth AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, Calgary, Canada, University of Calgary*, pages 1–6.

Compton, P. and Jansen, R. (1990). A philosophical Basis for Knowledge Acquisition. *Knowledge Kcquisition*, 2(3):241–258.

Compton, P., Preston, P., and Kang, B. (1995). The Use of Simulated Experts in Evaluating Knowledge Acquisition. In *Proceedings of the 9th Knowledge Acquisition for Knowledge-Based Systems Workshop*, pages 1–12.

Cruz, I. F., Antonelli, F. P., and Stroe, C. (2009). AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies. In *Proceedings of the VLDB Endowment*, volume 2, pages 1586–1589.

Cutler, B., Sheng, Y., Martin, S., Glaser, D., and Andersen, M. (2008). Interactive Selection of Optimal Fenestration Materials for Schematic Architectural Daylighting Design. *Automation in Construction*, 17(7):809–823.

Dhamankar, R., Lee, Y., Doan, A., Halevy, A., and Domingos, P. (2004). iMAP: Discovering Complex Semantic Matches between Database Schemas. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 383–394. ACM.

Djeddi, W. E. and Khadir, M. T. (2014). XMap++: Results for OAEI 2014. In *Proceedings of The 9th International Workshop on Ontology Matching Co-located with The 13th International Semantic Web Conference*, pages 163–169.

Do, H.-H., Melnik, S., and Rahm, E. (2003). Comparison of Schema Matching Evaluations. In *Web, Web-Services, and Database Systems*, pages 221–237. Springer.

Do, H.-H. and Rahm, E. (2002). COMA: A System for Flexible Combination of Schema Matching Approaches. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 610–621. VLDB Endowment.

Doan, A., Domingos, P., and Halevy, A. Y. (2001). Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach. In *ACM Sigmod Record*, volume 30, pages 509–520. ACM.

Doan, A. and Halevy, A. Y. (2005). Semantic Integration Research in the Database Community: A Brief Survey. *AI magazine, Special Issue on Semantic Integration*, 26(1):83–94.

Doan, A., Madhavan, J., Domingos, P., and Halevy, A. (2002). Learning to Map between Ontologies on the Semantic Web. In *Proceedings of the 11th International Conference on World Wide Web*, pages 662–673. ACM.

Dos Santos Mello, R. and Heuser, C. A. (2005). BInXS: A Process for Integration of XML Schemata. In *Advanced Information Systems Engineering*, pages 151–166. Springer.

Duchateau, F., Bellahsene, Z., and Coletta, R. (2008). A Flexible Approach for Planning Schema Matching Algorithms. In *On the Move to Meaningful Internet Systems: OTM 2008*, pages 249–264. Springer.

Duchateau, F., Coletta, R., Bellahsene, Z., and Miller, R. J. (2009). YAM: A Schema Matcher Factory. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 2079–2080. ACM.

Eckert, K., Meilicke, C., and Stuckenschmidt, H. (2009). Improving Ontology Matching Using Meta-Level Learning. In *The Semantic Web: Research and Applications*, pages 158–172. Springer.

Ehrig, M. and Sure, Y. (2004). Ontology Mapping - An Integrated Approach. In *The Semantic Web: Research and Applications*, pages 76–91. Springer.

Elmeleegy, H., Ouzzani, M., and Elmagarmid, A. (2008). Usage-based Schema Matching. In *Proceedings of the 24th International Conference on Data Engineering (ICDE)*, pages 20–29. IEEE.

Embley, D. W., Xu, L., and Ding, Y. (2004). Automatic Direct and Indirect Schema Mapping: Experiences and Lessons Learned. *ACM SIGMOD Record*, 33(4):14–19.

Feigenbaum, E. and McCorduck, P. (1983). *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*. Addison-Wesley Longman Publishing Co., Inc.

Frawley, W. J., Piatetsky-Shapiro, G., and Matheus, C. J. (1992). Knowledge Discovery in Databases: An Overview. *AI Magazine*, 13(3):57–70.

Freund, Y. and Mason, L. (1999). The Alternating Decision Tree Learning Algorithm. In *International Conference on Machine Learning (ICML)*, volume 99, pages 124–133.

Freund, Y. and Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. In *International Conference on Machine Learning (ICML)*, volume 96, pages 148–156.

Gaines, B. R. (1989). An Ounce of Knowledge is Worth a Ton of Data: Quantitative Studies of the Trade-Off between Expertise and Data Based on Statistically Well-Founded Empirical Induction. In *Machine Learning*, pages 156–159.

Gaines, B. R. and Compton, P. (1992). Induction of Ripple-Down Rules. In *Proceedings of the 5th Australian Conference on Artificial Intelligence*, volume 92, pages 349–354.

Gama, J. (2004). Functional Trees. *Machine Learning*, 55(3):219–250.

Giunchiglia, F. and Shvaiko, P. (2003). Semantic Matching. *Knowledge Engineering Review*, 18(3):265–280.

Glavic, B., Alonso, G., Miller, R. J., and Haas, L. M. (2010). TRAMP: Understanding the Behavior of Schema Mappings through Provenance. In *Proceedings of the VLDB Endowment*, volume 3, pages 1314–1325.

Guha, S., Rastogi, R., and Shim, K. (2000). ROCK: A Robust Clustering Algorithm for Categorical Attributes. *Information Systems*, 25(5):345–366.

Haas, L. M., Hernandez, M. A., Ho, H., Popa, L., and Roth, M. (2005). Clio Grows Up: from Research Prototype to Industrial Tool. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 805–810. ACM.

Hai, D. H. (2005). *Schema Matching and Mapping-Based Data Integration*. PhD thesis, Interdisciplinary Center for Bioinformatics and Department of Computer Science, University of Leipzig, Germany.

Hall, M. and Frank, E. (2008). Combining Naive Bayes and Decision Tables. In *Florida Artificial Intelligence Research Society (FLAIRS) Conference*, volume 2118, pages 318–319.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

Hartung, M., Terwilliger, J., and Rahm, E. (2011). Recent Advances in Schema and Ontology Evolution. In *Schema Matching and Mapping*, pages 149–190. Springer.

Hoshiai, T., Yamane, Y., Nakamura, D., and Tsuda, H. (2004). A Semantic Category Matching Approach to Ontology Alignment. In *Proceedings of The 3rd International Workshop on The Evaluation of Ontology-Based Tools*.

Hu, W., Qu, Y., and Cheng, G. (2008). Matching Large Ontologies: A Divide-And-Conquer Approach. *Data and Knowledge Engineering*, 67(1):140–160.

Ichise, R. (2008). Machine Learning Approach for Ontology Mapping Using Multiple Concept Similarity Measures. In *7th IEEE/ACIS International Conference on Computer and Information Science (ICIS)*, pages 340–346. IEEE.

Jain, P., Hitzler, P., Sheth, A. P., Verma, K., and Yeh, P. Z. (2010a). Ontology Alignment for Linked Open Data. In *The Semantic Web - ISWC 2010*, volume 6496, pages 402–417. Springer.

Jain, P., Hitzler, P., Yeh, P. Z., Verma, K., and Sheth, A. P. (2010b). Linked Data Is Merely More Data. In *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*, volume 11, pages 82–86.

Jean-Mary, Y. R., Shironoshita, E. P., and Kabuka, M. R. (2009). Ontology Matching with Semantic Verification. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):235–251.

Jimenez, S., Becerra, C., Gelbukh, A., and Gonzalez, F. (2009). Generalized Mongue-Elkan Method for Approximate Text String Comparison. In *Computational Linguistics and Intelligent Text Processing*, pages 559–570. Springer.

Jimenez-Ruiz, E. and Grau, B. C. (2011). LogMap: Logic-Based and Scalable Ontology Matching. In *The Semantic Web - ISWC 2011*, pages 273–288. Springer.

Jimenez-Ruiz, E., Grau, B. C., Zhou, Y., and Horrocks, I. (2012). Large-Scale Interactive Ontology Matching: Algorithms and Implementation. In *European Conference on Artificial Intelligence (ECAI)*, pages 444–449.

Kang, B., Compton, P., and Preston, P. (1995). Multiple Classification Ripple Down Rules: Evaluation and Possibilities. In *Proceedings of the 9th Knowledge Acquisition for Knowledge-Based Systems Workshop*, volume 1, pages 17.1–17.20.

Kang, B. H. (1996). *Validating Knowledge Acquisition: Multiple Classification Ripple-Down Rules*. PhD thesis, University of New South Wales.

Kang, B. H. and Compton, P. (1994). A Maintenance Approach to Case Based Reasoning. In *Advances in Case-Based Reasoning, 2nd European Workshop, EWCBR-94*, volume 984, pages 226–239. Springer.

Khiat, A. and Benaissa, M. (2014). AOT/AOTL Results for OAEI 2014. In *Proceedings of the 9th International Workshop on Ontology Matching Co-located with the 13th International Semantic Web Conference (ISWC 2014)*, volume 1317, pages 113–119. CEUR-WS. org.

Kim, Y. S., Compton, P., and Kang, B. H. (2012). Ripple-Down Rules with Censored Production Rules. In *Knowledge Management and Acquisition for Intelligent Systems*, pages 175–187. Springer.

Kirsten, T., Gross, A., Hartung, M., and Rahm, E. (2011). GOMMA: A Component-Based Infrastructure for Managing and Analyzing Life Science Ontologies and Their Evolution. *Journal of Biomedical Semantics*, 2(6).

Kirsten, T., Thor, A., and Rahm, E. (2007). Instance-Based Matching of Large Life Science Ontologies. In *Data Integration in the Life Sciences*, pages 172–187. Springer.

Kohavi, R. (1996). Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision Tree Hybrid. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, volume 96, pages 202–207. AAAI Press.

Koudas, N., Sarawagi, S., and Srivastava, D. (2006). Record Linkage: Similarity Measures and Algorithms. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 802–803. ACM.

Lambrix, P. and Tan, H. (2006). SAMBO - A System for Aligning and Merging Biomedical Ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(3):196–206.

Lee, J. M. and Andersen, M. (2009). A Simulation-Based Expert System for Daylighting Design. *Proceedings of Lux Europa*.

Lee, Y., Sayyadian, M., Doan, A., and Rosenthal, A. S. (2007). eTuner: Tuning Schema Matching Software Using Synthetic Scenarios. *The VLDB Journal-The International Journal on Very Large Data Bases*, 16:97–122.

Lenzerini, M. (2002). Data Integration: A Theoretical Perspective. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 233–246. ACM.

Li, J., Tang, J., Li, Y., and Luo, Q. (2009). RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. *Knowledge and Data Engineering, IEEE Transactions*, 21(8):1218–1232.

Li, W.-S. and Clifton, C. (2000). SEMINT: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural Networks. *Data and Knowledge Engineering*, 33(1):49–84.

Loh, W.-Y. (2011). Classification and Regression Trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23.

Madhavan, J., Bernstein, P. A., Doan, A., and Halevy, A. (2005). Corpus-Based Schema Matching. In *Proceedings of the 21st International Conference on Data Engineering (ICDE)*, pages 57–68. IEEE.

Madhavan, J., Bernstein, P. A., and Rahm, E. (2001). Generic Schema Matching with Cupid. In *Proceeding of the 27th International Conference On Very Large Data Bases (VLDB)*, volume 1, pages 49–58.

Maedche, A. and Staab, S. (2002). Measuring Similarity between Ontologies. In *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 251–263. Springer.

Mao, M., Peng, Y., and Spring, M. (2010). An Adaptive Ontology Mapping Approach with Neural Network Based Constraint Satisfaction. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(1):14–25.

Marie, A. and Gal, A. (2008). Boosting Schema Matchers. In *On the Move to Meaningful Internet Systems: OTM 2008*, pages 283–300. Springer.

Martinez-Gil, J., Navas-Delgado, I., and Aldana-Montes, J. F. (2012). MaF: An Ontology Matching Framework. *Journal of Universal Computer Science*, 18(2):194–217.

Massmann, S. and Rahm, E. (2008). Evaluating Instance-Based Matching of Web Directories. In *International Workshop on the Web and Databases (WebDB)*.

McBrien, P. and Poulovassilis, A. (2003). Data Integration by Bi-Directional Schema Transformation Rules. In *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, pages 227–238. IEEE.

Mecca, G., Papotti, P., and Raunich, S. (2009). Core Schema Mappings. In *Proceedings of the ACM SIGMOD International Conference on Management of data*, pages 655–668. ACM.

Meilicke, C. (2011). *Alignment Incoherence in Ontology Matching*. PhD thesis, University of Mannheim.

Melnik, S., Garcia-Molina, H., and Rahm, E. (2002). Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *Proceedings of the 18th International Conference on Data Engineering (ICDE)*, pages 117–128. IEEE.

Melnik, S., Rahm, E., and Bernstein, P. A. (2003). Rondo: A Programming Platform for Generic Model Management. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 193–204. ACM.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Milo, T. and Zohar, S. (1998). Using Schema Matching to Simplify Heterogeneous Data Translation. In *Proceedings of the 24th International Conference On Very Large Data Bases (VLDB)*, volume 98, pages 24–27.

Mitra, P., Wiederhold, G., and Jannink, J. (1999). Semi-Automatic Integration of Knowledge Sources. In *Proceedings of the 2nd International Conference on Information Fusion*.

Mulholland, M., Preston, P., Sammut, C., Hibbert, B., and Compton, P. (1993). An Expert System for Ion Chromatography Developed Using Machine Learning and Knowledge in Context. In *Proceedings of the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 258–267. Gordon and Breach Science Publishers.

Nandi, A. and Bernstein, P. A. (2009). HAMSTER: Using Search Clicklogs for Schema and Taxonomy Matching. In *Proceedings of the VLDB Endowment*, volume 2, pages 181–192.

Ngo, D., Bellahsene, Z., and Coletta, R. (2011a). A Generic Approach for Combining Linguistic and Context Profile Metrics in Ontology Matching. In *On the Move to Meaningful Internet Systems: OTM*, pages 800–807. Springer.

Ngo, D., Bellahsene, Z., and Todorov, K. (2013). Opening the Black Box of Ontology Matching. In *The Semantic Web: Semantics and Big Data*, pages 16–30. Springer.

Ngo, D. H. and Bellahsene, Z. (2012). YAM++:(not) Yet Another Matcher for Ontology Matching Task. In *BDA*.

Ngo, D. H., Bellahsene, Z., and Coletta, R. (2011b). Yam++–results for oaei 2011. In *ISWC'11: The 6th International Workshop on Ontology Matching*, volume 814, pages 228–235.

Noy, N. F. and Klein, M. (2004). Ontology Evolution: Not the Same as Schema Evolution. *Knowledge and Information Systems*, 6(4):428–440.

Peukert, E., Eberius, J., and Rahm, E. (2011). AMC-A Framework for Modelling and Comparing Matching Systems as Matching Processes. In *Proceedings of the 27th International Conference on Data Engineering (ICDE)*, pages 1304–1307. IEEE.

Peukert, E., Eberius, J., and Rahm, E. (2012). A Self-Configuring Schema Matching System. In *Proceedings of the 28th International Conference on Data Engineering (ICDE)*, pages 306–317. IEEE.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*, volume 16, pages 235–240. Morgan Kaufmann Publishers Inc.

Rahm, E. (2011). Towards Large-Scale Schema and Ontology Matching. In *Schema Matching and Mapping*, pages 3–27. Springer.

Rahm, E. and Bernstein, P. A. (2001). A Survey of Approaches to Automatic Schema Matching. *The VLDB Journal*, 10(4):334–350.

Ramanath, M., Freire, J., Haritsa, J. R., and Roy, P. (2003). Searching for Efficient XML-to-Relational Mappings. In *Database and XML Technologies*, pages 19–36. Springer.

Richards, D. (2009). Two Decades of Ripple Down Rules Research. *The Knowledge Engineering Review*, 24(02):159–184.

Russell, S., Norvig, P., and Intelligence, A. (1995). A Modern Approach. *Artificial Intelligence, Prentice-Hall, Egnlewood Cliffs*, 25.

Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998). A Bayesian Approach to Filtering Junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, volume 62, pages 98–105.

Santos, E., Faria, D., Pesquita, C., and Couto, F. (2015). Ontology Alignment Repair Through Modularization and Confidence-Based Heuristics. *PloS one*, 10(12).

Sayyadian, M., Lee, Y., Doan, A., and Rosenthal, A. S. (2005). Tuning Schema Matching Software Using Synthetic Scenarios. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 994–1005. VLDB Endowment.

Schadd, F. C. and Roos, N. (2014). Alignment Evaluation of MaasMatch for the OAEI 2014 Campaign. In *Proceedings of the 9th International Workshop on Ontology Matching Co-located with the 13th International Semantic Web Conference (ISWC 2014)*.

Seddiqui, M. H. and Aono, M. (2009). An Efficient and Scalable Algorithm for Segmented Alignment of Ontologies of Arbitrary Size. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(4):344–356.

Shen, G., Liu, Y., Fei Wang, J. S., Wang, Z., Huang, Z., and Kang, D. (2014). OM-Reasoner: Combination of Multi-matchers for Ontology Matching: Results for OAEI 2014. In *Proceedings of the 9th International Conference on Ontology Matching*, volume 1317, pages 142–148. CEUR-WS. org.

Shvaiko, P. and Euzenat, J. (2005). A Survey of Schema-Based Matching Approaches. *Journal on Data Semantics IV*, pages 146–171.

Shvaiko, P. and Euzenat, J. (2013). Ontology Matching: State of the Art and Future Challenges. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):158–176.

Stoilos, G., Stamou, G., and Kollias, S. (2005). A string Metric for Ontology Alignment. In *The Semantic Web-ISWC 2005*, pages 624–637. Springer.

Thor, A., Kirsten, T., and Rahm, E. (2007). Instance-based Matching of Hierarchical Ontologies. In *International Board Test Workshop (BTW)*, volume 103, pages 436–448.

Tsoumakas, G. and Katakis, I. (2007). Multi-Label Classification: An Overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13.

Uschold, M. and Gruninger, M. (2004). Ontologies and Semantics for Seamless Connectivity. *ACM SIGMOD Record*, 33(4):58–64.

Volz, J., Bizer, C., Gaedke, M., and Kobilarov, G. (2009). Discovering and Maintaining Links on The Web of Data. In *Internatioanl Semantic Web Conference (ISWC)*, volume 5823. Springer.

Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hubner, S. (2001). Ontology-Based Integration of Information - A Survey of

Existing Approaches. In *IJCAI-01 Workshop: Ontologies and Information Sharing*, volume 2001, pages 108–117.

Wang, P. (2011). Lily Results on SEALS Platform for OAEI 2011. In *Proceedings of the 6th OM Workshop*, pages 156–162.

Wang, P. and Xu, B. (2009). An Effective Similarity Propagation Method for Matching Ontologies without Sufficient or Regular Linguistic Information. *The Semantic Web*, pages 105–119.

Wimalasuriya, D. C. and Dou, D. (2010). Ontology-Based Information Extraction: An Introduction and a Survey of Current Approaches. *Journal of Information Science*, 36(3):306–323.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*, chapter 1,11, pages 38,447,456. Morgan Kaufmann.

Xu, H. (2011). *RDRCE - A Framework for the Efficient Construction of Knowledge Bases for NLP Tasks*. PhD thesis, University of New South Wales.

Yu, C. and Popa, L. (2005). Semantic Adaptation of Schema Mappings When Schemas Evolve. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 1006–1017. VLDB Endowment.

Zhong, Q., Li, H., Li, J., Xie, G., Tang, J., Zhou, L., and Pan, Y. (2009). A Gauss Function Based Approach for Unbalanced Ontology Matching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 669–680. ACM.